



**University of
Zurich**^{UZH}

**Zurich Open Repository and
Archive**

University of Zurich
University Library
Strickhofstrasse 39
CH-8057 Zurich
www.zora.uzh.ch

Year: 2019

Microbial residence time is a controlling parameter of the taxonomic composition and functional profile of microbial communities

Mansfeldt, Cresten ; Achermann, Stefan ; Men, Yujie ; Walser, Jean-Claude ; Villez, Kris ; Joss, Adriano ; Johnson, David R ; Fenner, Kathrin

Abstract: A remaining challenge within microbial ecology is to understand the determinants of richness and diversity observed in environmental microbial communities. In a range of systems, including activated sludge bioreactors, the microbial residence time (MRT) has been previously shown to shape the microbial community composition. However, the physiological and ecological mechanisms driving this influence have remained unclear. Here, this relationship is explored by analyzing an activated sludge system fed with municipal wastewater. Using a model designed in this study based on Monod-growth kinetics, longer MRTs were shown to increase the range of growth parameters that enable persistence, resulting in increased richness and diversity in the modeled community. In laboratory experiments, six sequencing batch reactors treating domestic wastewater were operated in parallel at MRTs between 1 and 15 days. The communities were characterized using both 16S ribosomal RNA and non-target messenger RNA sequencing (metatranscriptomic analysis), and model-predicted monotonic increases in richness were confirmed in both profiles. Accordingly, taxonomic Shannon diversity also increased with MRT. In contrast, the diversity in enzyme class annotations resulting from the metatranscriptomic analysis displayed a non-monotonic trend over the MRT gradient. Disproportionately high abundances of transcripts encoding for rarer enzymes occur at longer MRTs and lead to the disconnect between taxonomic and functional diversity profiles.

DOI: <https://doi.org/10.1038/s41396-019-0371-6>

Posted at the Zurich Open Repository and Archive, University of Zurich

ZORA URL: <https://doi.org/10.5167/uzh-179739>

Journal Article

Accepted Version

Originally published at:

Mansfeldt, Cresten; Achermann, Stefan; Men, Yujie; Walser, Jean-Claude; Villez, Kris; Joss, Adriano; Johnson, David R; Fenner, Kathrin (2019). Microbial residence time is a controlling parameter of the taxonomic composition and functional profile of microbial communities. The ISME journal, 13(6):1589-1601.

DOI: <https://doi.org/10.1038/s41396-019-0371-6>

Supplementary Figures

Table of Contents

Supplemental Figure 1. The distribution of 16S reads per sequenced libraries (in triplicate) for the two TPs and two primers (B1 and B2). The [+] samples were positive mock communities used to assess both the accuracy of the negative signal and the recovery of an expected distribution. The [-] samples were blanks carried through the PCR and sequencing steps. Note: TP2 B17d gDNA 2 displayed poor performance, and TP1 B2 [+] gDNA 3 is not displayed because that sample library contained ten-times the reads when compared to the other TP1 B2 samples.

Supplemental Figure 2. Rarefaction curves for the 16S rRNA data. The number of resulting ESVs from triplicate libraries for the rRNA (solid) and rDNA (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors are plotted against the sub-selection read count for both Primer B1 and B2. The positive mock communities are plotted in grey.

Supplemental Figure 3. Tape Station traces of the RNA pool before rRNA depletion. The numbers below the figure indicate the MRT of the reactor, ladder (+), and empty control (-). The numbers above the figure indicate the RNA Integrity Number (RIN).

Supplemental Figure 4. Bioanalyzer traces of the RNA pool after rRNA depletion using the RiboZero Epidemiology Kit. RIN indicates the RNA Integrity Number.

Supplemental Figure 5. Fragment quality of the library run on the Illumina NextSEQ.

Supplemental Figure 6. TP1 48 day raw FastQC quality results.

Supplemental Figure 7. TP2 187 day raw FastQC quality results.

Supplemental Figure 8. TP1 48 day trimmed FastQC quality results.

Supplemental Figure 9. TP2 187 day trimmed FastQC quality results.

Supplemental Figure 10. Bitscore histograms representing the quality of the DIAMOND annotation for all reactors against the Uniprot EC library only (14,871,396 sequences, downloaded on March 6th, 2018).

Supplemental Figure 11. Bitscore histograms representing the quality of the DIAMOND with a bitscore cutoff of 50 annotation for all reactors against the Uniprot EC library (14,871,396 sequences, downloaded on March 6th, 2018).

Supplemental Figure 12. Rarefaction curves for the EC number annotations for the TP1 (solid) and TP2 (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors.

Supplemental Figure 13. Bitscore histograms representing the quality of the DIAMOND annotation for TP1R1 with default DIAMOND values against the (a) Uniprot EC library only (14,871,396 sequences, 5,767,210,689 amino acids, with an average of 387.8 a.a., downloaded on March 6th, 2018) and (b) the full Uniprot library (109,414,541 sequences, 36,814,708,888 amino acids, average length of protein 336.5 a.a., downloaded on March 6th, 2018).

Supplemental Figure 14. The range of μ_{\max} and K_s parameters for the 1 and 15 d MRT Sequencing Batch Reactor models when allowing only the μ_{\max} and K_s parameters to vary (the b_e is held constant). For comparison, see Figure 1.b within the main text.

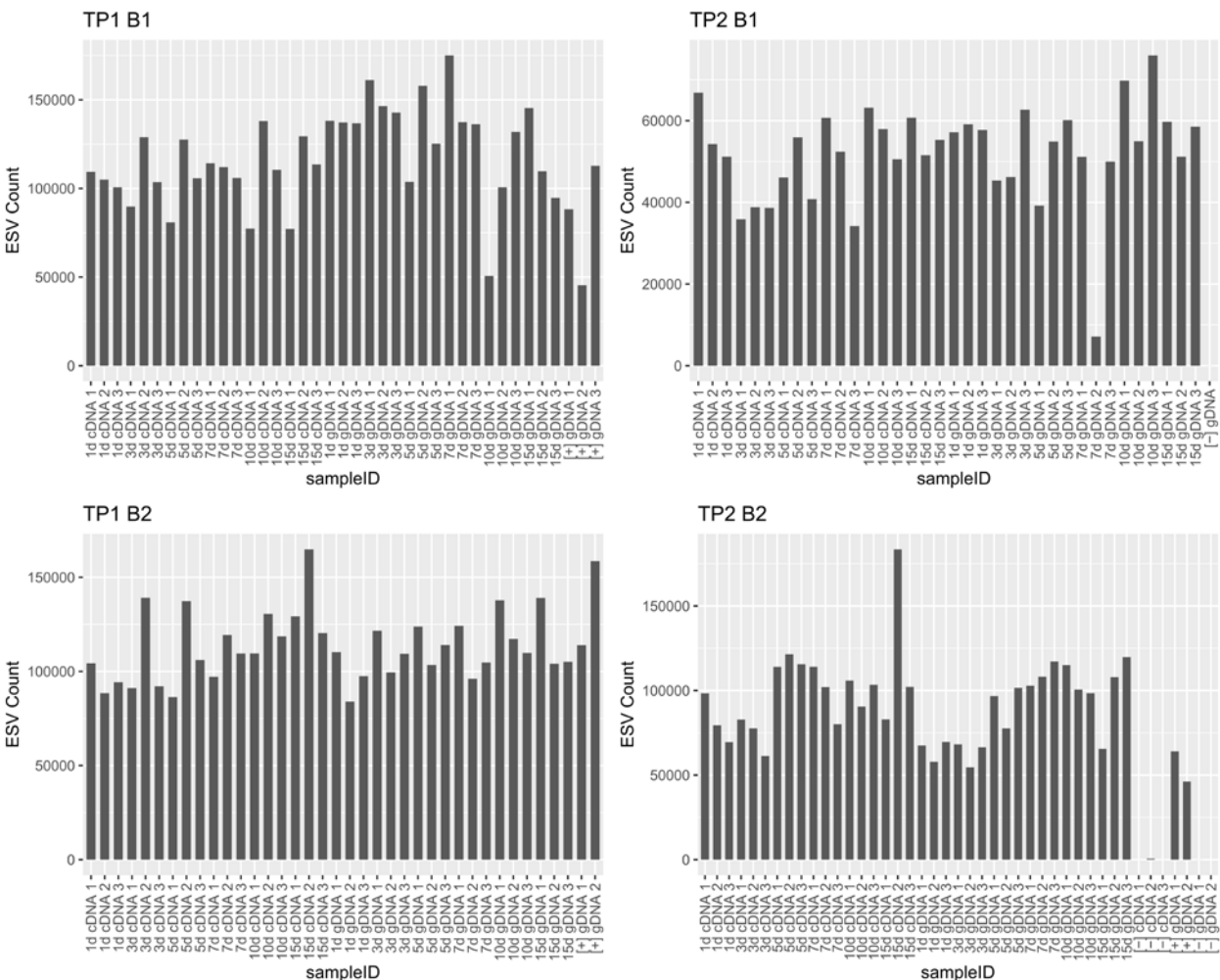
Supplemental Figure 15. A trace of the growth parameter solution path, the Shannon diversity, and the richness results of the model when the of μ_{\max} and b_e are constrained (a) by the fastest grower or (b) by the slowest grower.

Supplemental Figure 16. (a,b) The calculated Hill numbers (0-3) for the rarified B1-primer amplified 16S rRNA (black) and rDNA (red) data. (c,d) The abundance data distributed into taxonomic orders; the top 10 of the sums across each time-point are colored, resulting in 15 orders being represented.

Supplementary Figure 17. Summary of all EC sub-subclasses across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and ward clustering of the (a) EC sub-subclass fraction of total abundance. (b) The 15/1d MRT abundance log ratio. The within EC sub-subclass taxonomic (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC sub-subclass originating from a Eukaryotic sequence in the Uniprot database.

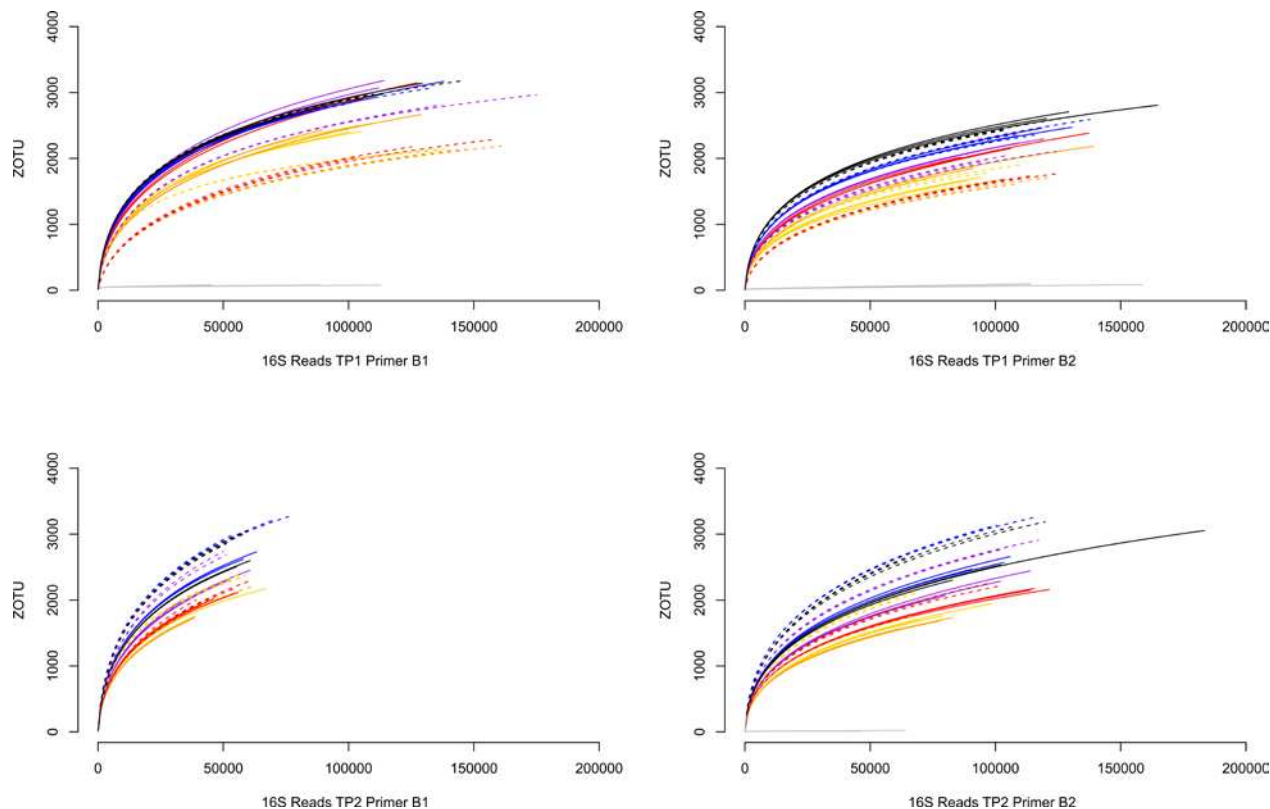
Supplementary Figure 18. Summary of the top 50 maximum fractional abundances EC numbers across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and ward clustering of the (a) EC number fraction of total abundance. (b) The 15/1d MRT abundance log ratio. The within EC number taxonomic (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC number originating from a Eukaryotic sequence in the Uniprot database.

Supplementary Figure 19. The Hill 0 (a) and 1 (b) values for the functionally annotated RNA originating from Bacteria only data binned into common EC number for TP1 (solid) and TP2 (dashed) at the fractional abundance cutoff of 10^{-7} . The Hill 0 and 1 values were calculated for each reactor using the rtk v0.2.5.4 package in R v3.5.1 for 10 bootstrap sub-selections of the annotations. The lines trace the bootstrap mean.



Supplemental Figure 1. The distribution of 16S reads per sequenced libraries (in triplicate) for the two TPs and two primers (B1 and B2). The [+] samples were positive mock communities used to assess both the accuracy of the negative signal and the recovery of an expected distribution. The [-] samples were blanks carried through the PCR and sequencing steps. Note: only TP2 B1 7d gDNA 2 displayed poor performance, and TP1 B2 [+] gDNA 3 is not displayed because that sample library contained ten-times the reads when compared to the other TP1 B2 samples.

100

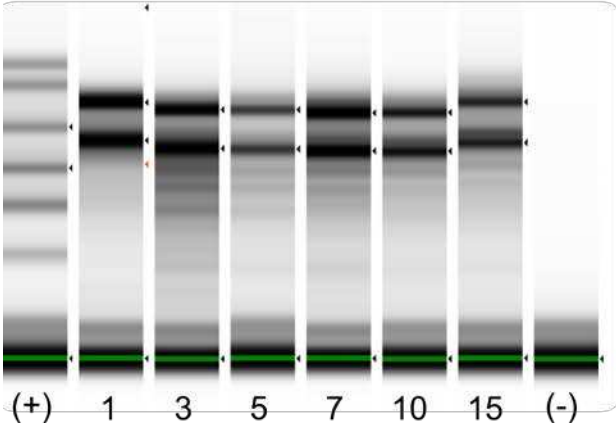


101
102
103
104
105

Supplemental Figure 2. Rarefaction curves for the 16S rRNA data. The number of resulting ESVs from triplicate libraries for the 16S rRNA (solid) and rDNA (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors are plotted against the sub-selection read count for both Primer B1 and B2. The positive mock communities are plotted in grey.

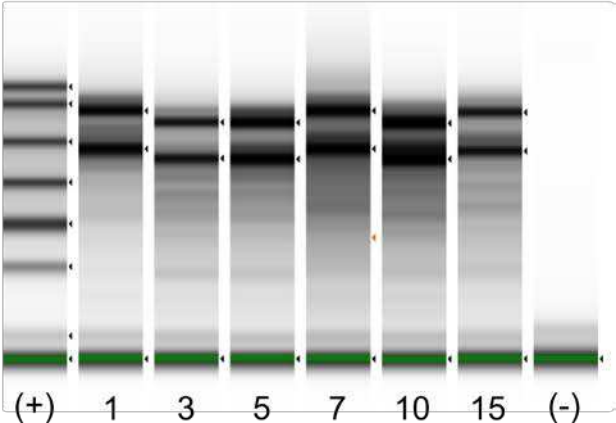
TP1 48 days

RIN 7.7 5.9 6.1 7.3 6.9 6.7

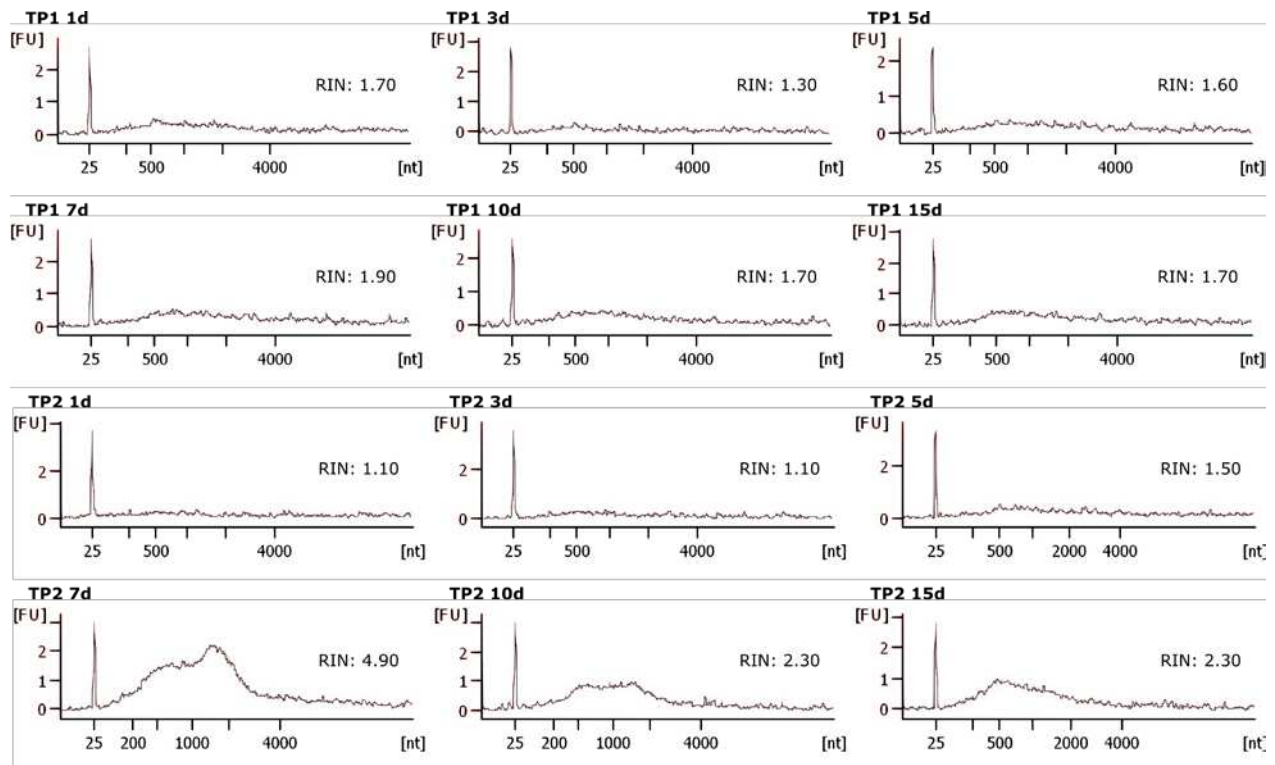


TP2 187 days

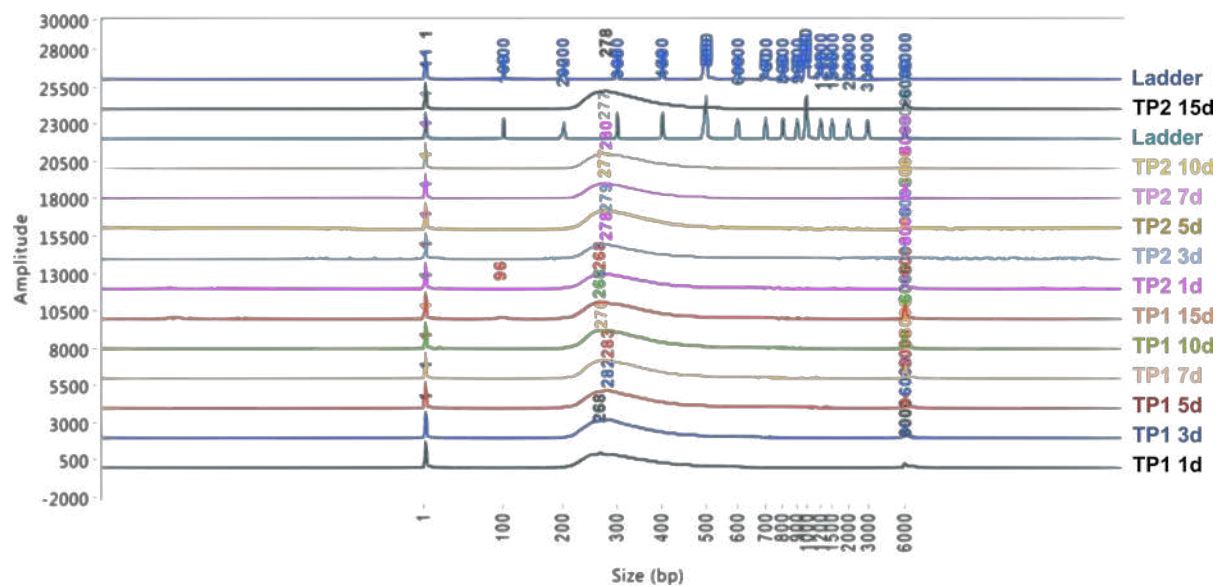
7.1 6.0 6.5 5.9 6.7 6.6



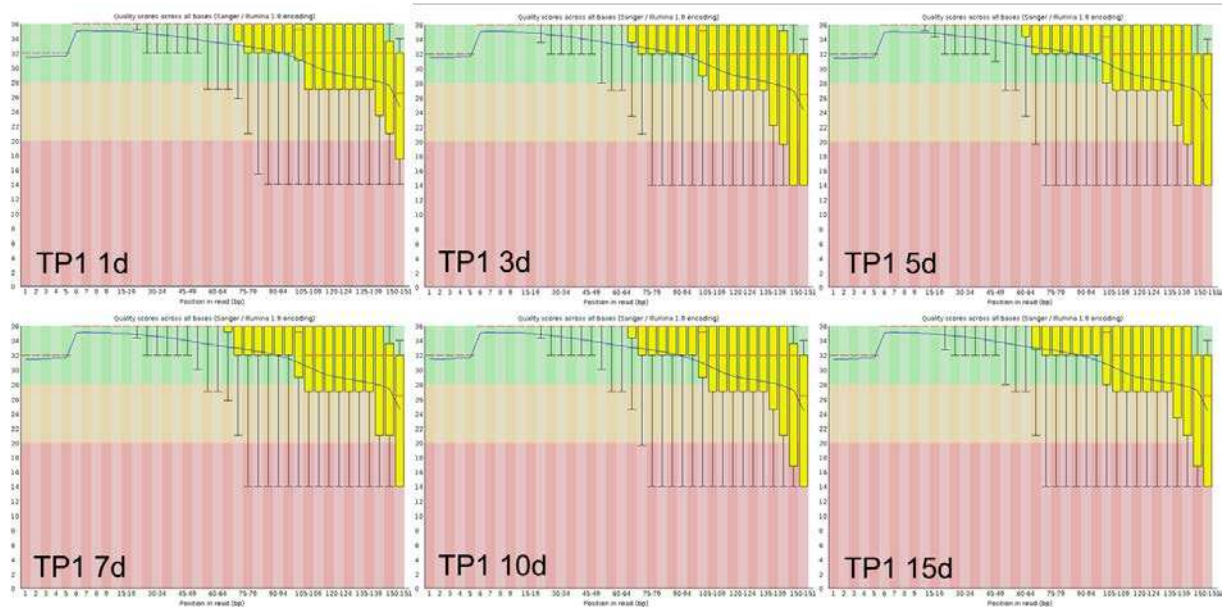
Supplemental Figure 3. Tape Station traces of the RNA pool before rRNA depletion. The numbers below the figure indicate the MRT of the reactor, ladder (+), and empty control (-). The numbers above the figure indicate the RNA Integrity Number (RIN).



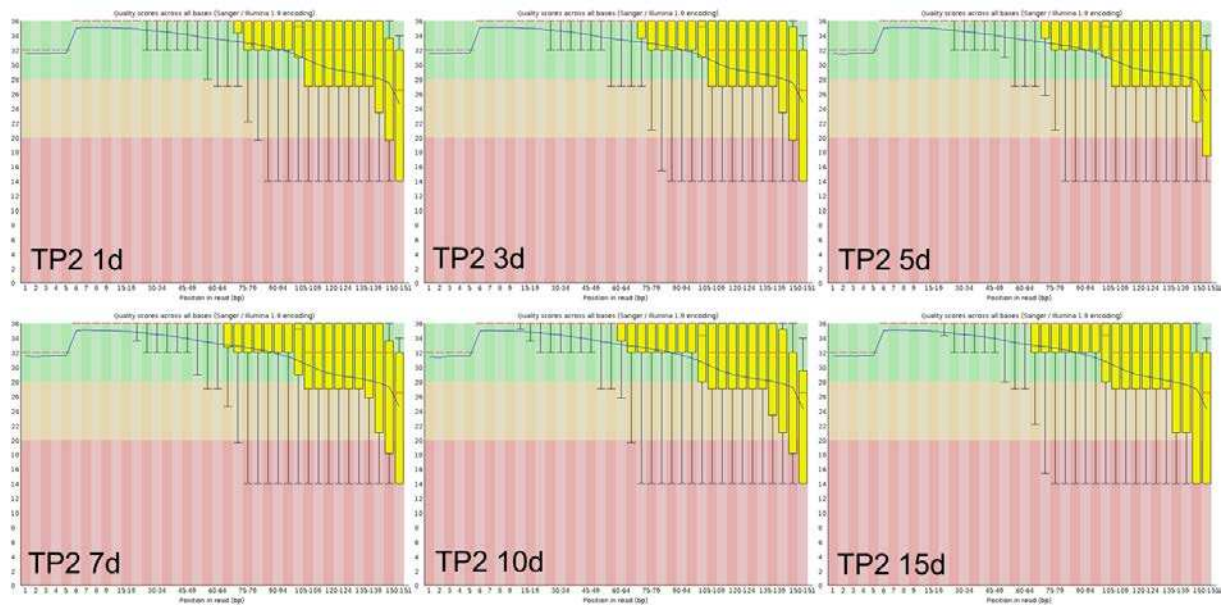
Supplemental Figure 4. Bioanalyzer traces of the RNA pool after rRNA depletion using the RiboZero Epidemiology Kit. RIN indicates the RNA Integrity Number.



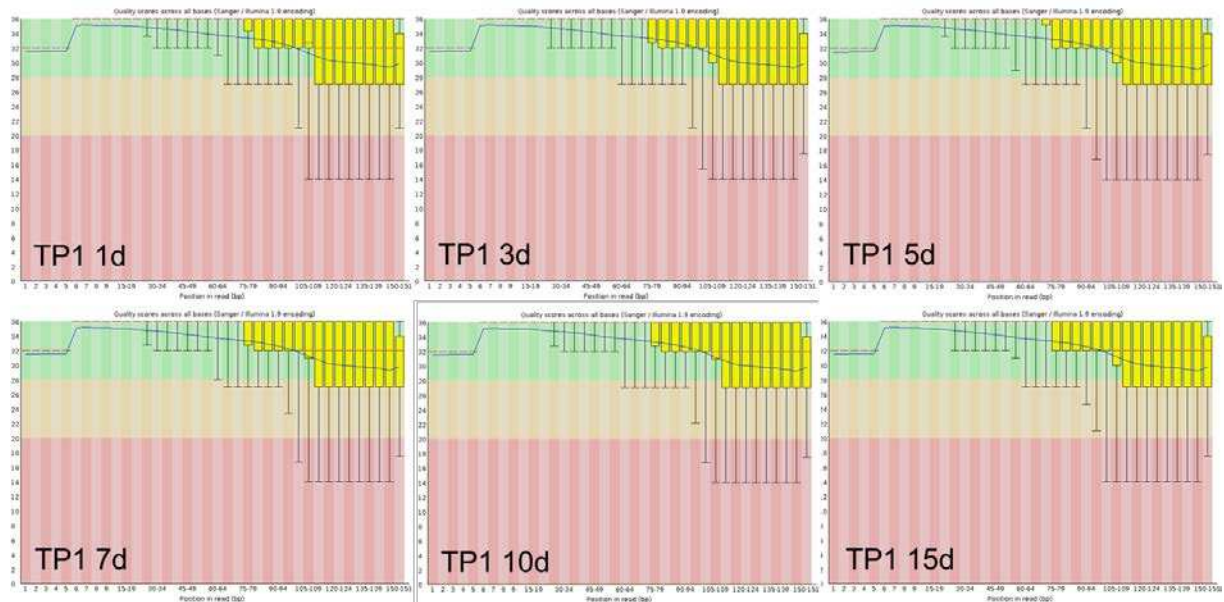
Supplemental Figure 5. Fragment quality of the library run on the Illumina NextSEQ.



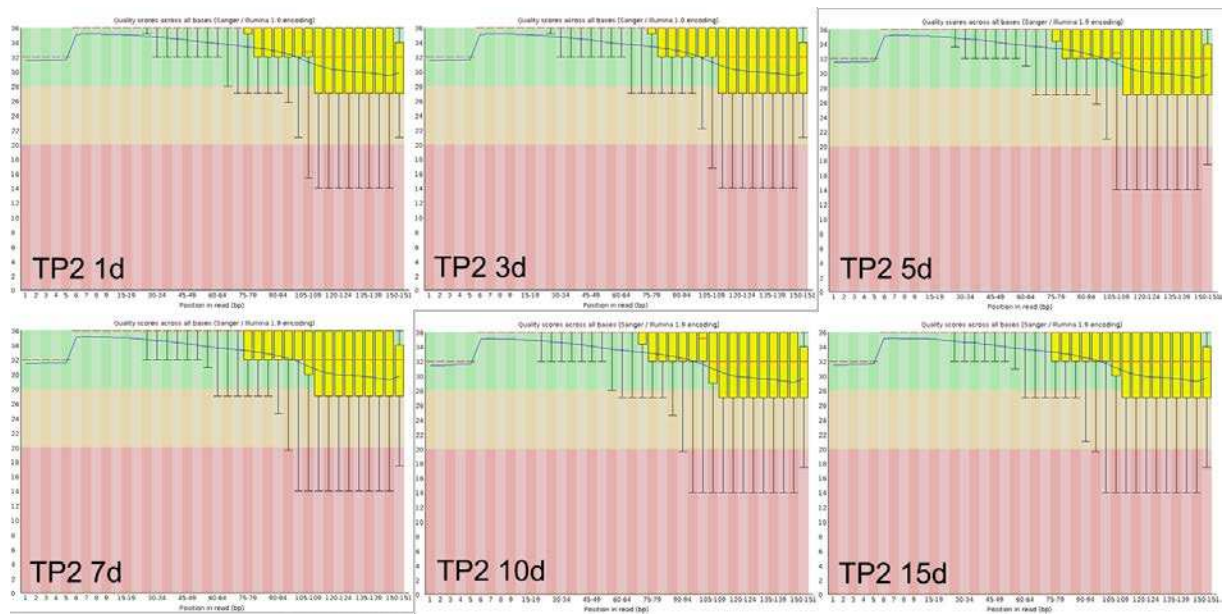
Supplemental Figure 6. TP1 48 day raw FastQC quality results.



Supplemental Figure 7. TP2 187 day raw FastQC quality results.

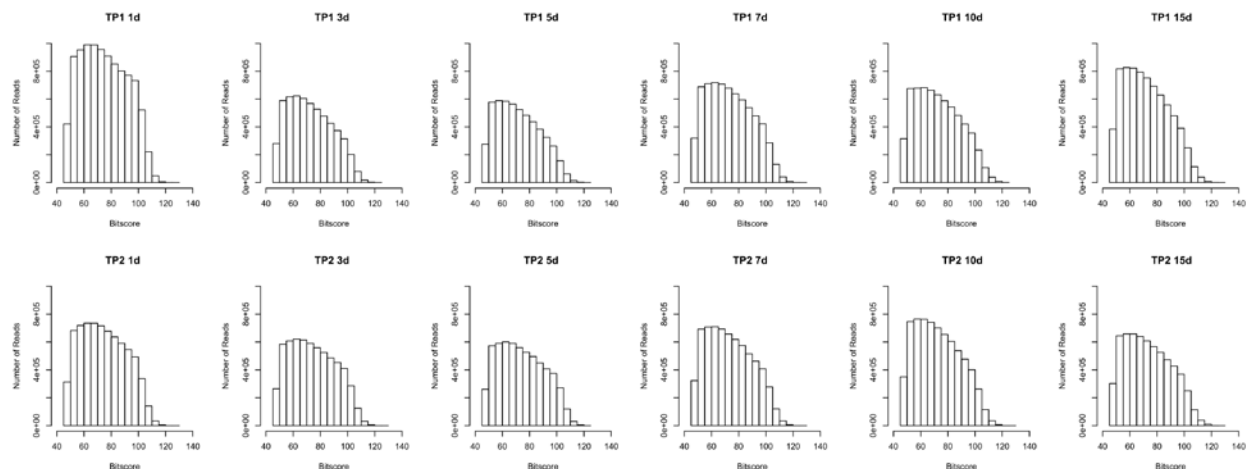


Supplemental Figure 8. TP1 48 day trimmed FastQC quality results.



Supplemental Figure 9. TP2 187 day trimmed FastQC quality results.

131



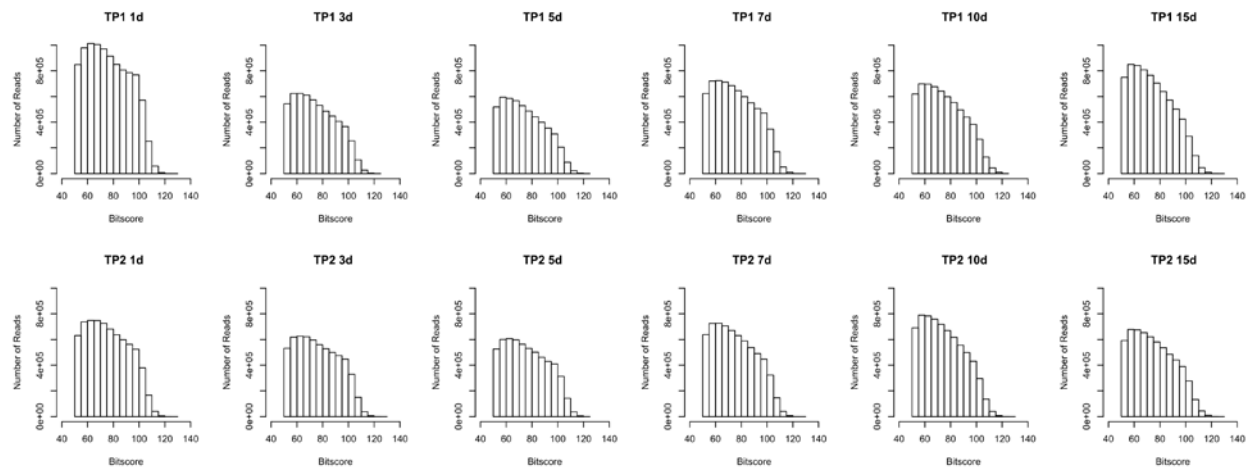
132

133

134

135

Supplemental Figure 10. Bitscore histograms representing the quality of the DIAMOND annotation for all reactors against the Uniprot EC library only (14,871,396 sequences, downloaded on March 6th, 2018).



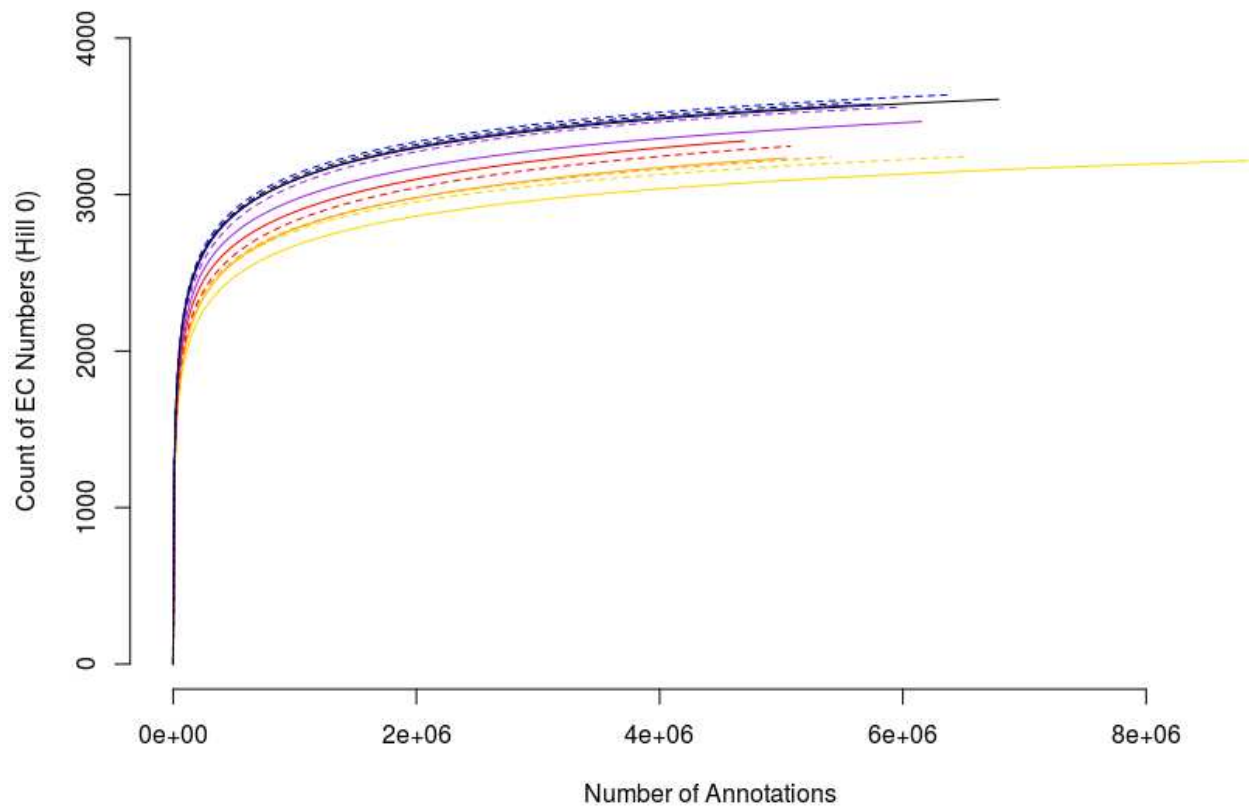
136

137

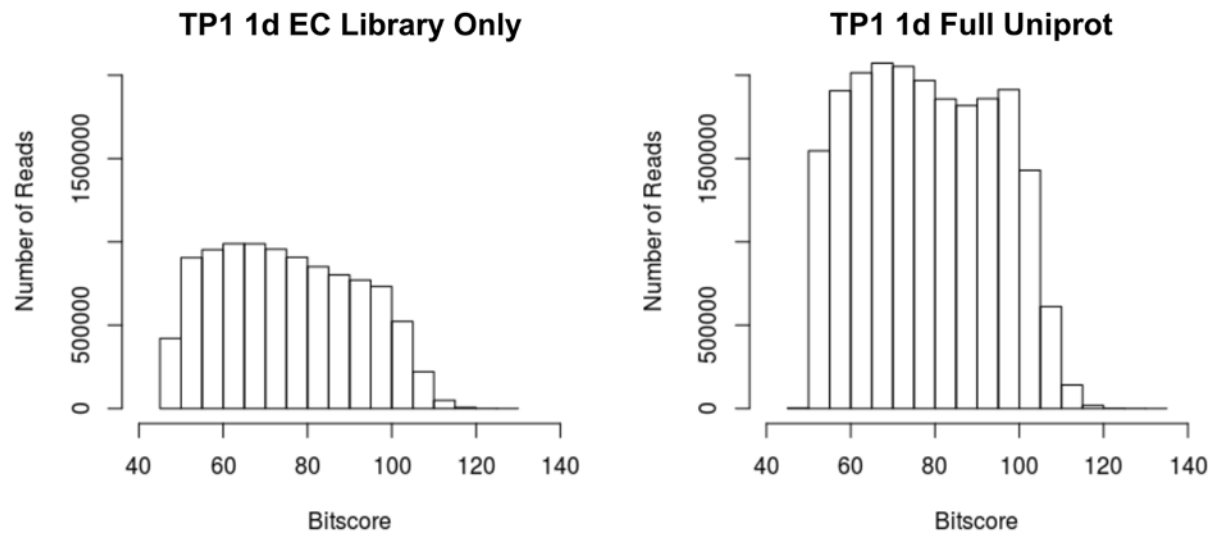
138

139

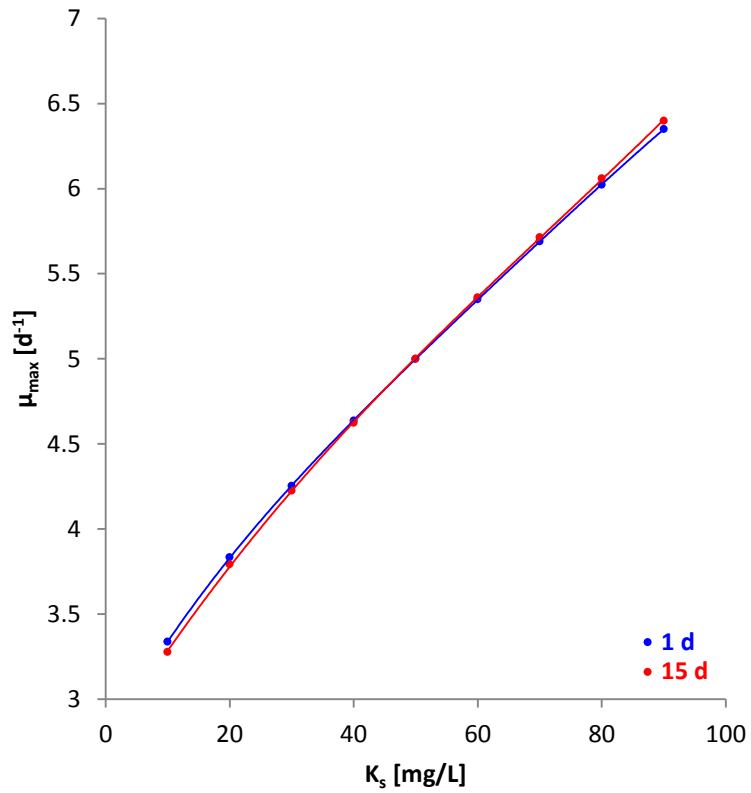
Supplemental Figure 11. Bitscore histograms representing the quality of the DIAMOND with a bitscore cutoff of 50 annotation for all reactors against the Uniprot EC library (14,871,396 sequences, downloaded on March 6th, 2018).



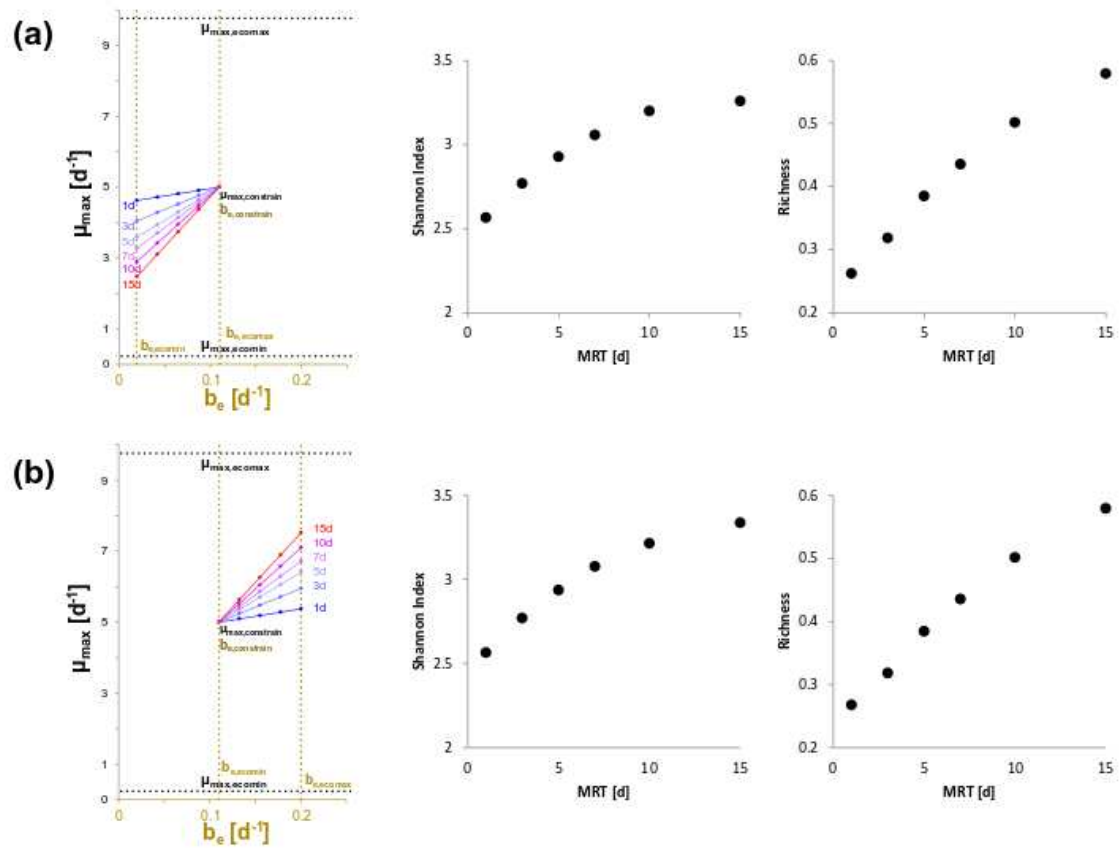
Supplemental Figure 12. Rarefaction curves for the EC number annotations for the TP1 (solid) and TP2 (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors.



Supplemental Figure 13. Bitscore histograms representing the quality of the DIAMOND annotation for TP1 1 d MRT with default DIAMOND values against the (a) Uniprot EC library only (14,871,396 sequences, 5,767,210,689 amino acids, with an average of 387.8 a.a., downloaded on March 6th, 2018) and (b) the full Uniprot library (109,414,541 sequences, 36,814,708,888 amino acids, average length of protein 336.5 a.a., downloaded on March 6th, 2018).

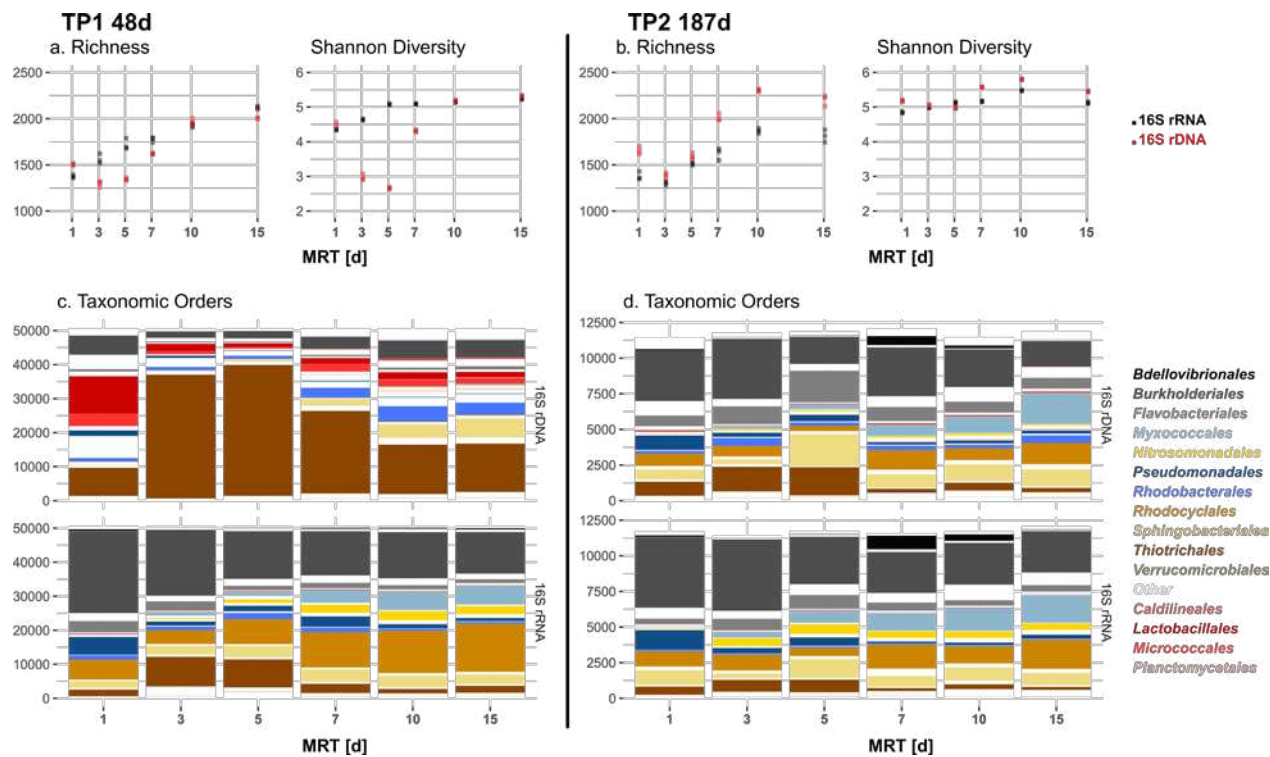


Supplemental Figure 14. The range of μ_{max} and K_s parameters for the 1 and 15 d MRT Sequencing Batch Reactor models when allowing only the μ_{max} and K_s parameters to vary (the b_e is held constant). For comparison, see Figure 1.b within the main text.



Supplemental Figure 15. A trace of the growth parameter solution path, the Shannon diversity, and the richness results of the model when the of μ_{\max} and b_e are constrained (a) by the fastest grower or (b) by the slowest grower.

157



158

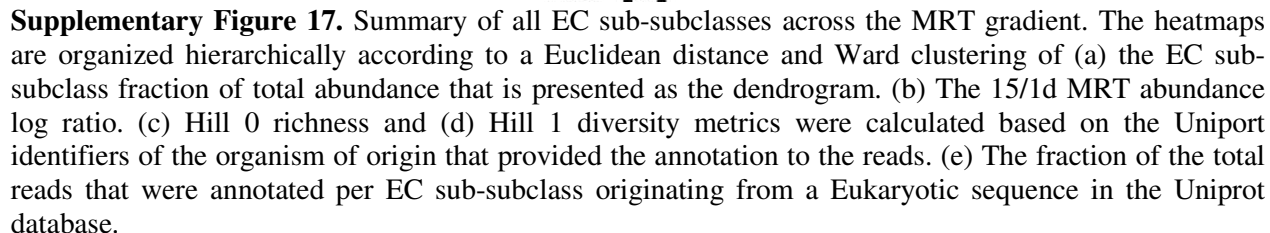
159

160

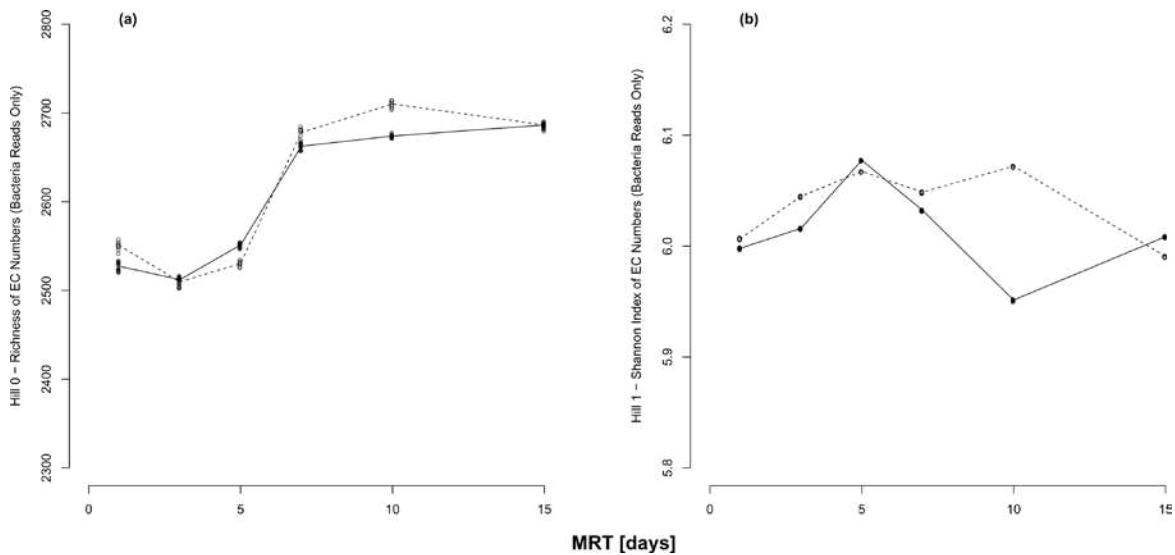
161

162

Supplemental Figure 16. a,b. The calculated richness and Shannon diversity for the B1-primer amplified 16S rRNA (black) and rDNA (light blue) data. c,d. The abundance data distributed into taxonomic orders; the top 10 of the sums across each experiment are colored, resulting in 12 orders being represented.



183



184
185
186
187
188
189
190

Supplementary Figure 19. The richness Hill₀ (a) and Shannon Diversity Hill₁ (b) values for the functionally annotated RNA originating from Bacteria only data binned into common EC numbers for TP1 (solid) and TP2 (dashed) at the fractional abundance cutoff of 10⁻⁷. The Hill 0 and 1 values were calculated for each reactor using the rtk package in R for 10 bootstrap sub-selections of the annotations. The lines trace the bootstrap mean.

191	Supplementary Files
192	
193	Table of Contents
194	
195	Supplemental File 1 : Phyloseq processing of the 16S rRNA reads
196	
197	Supplemental File 2 : In silico quality filtering, rRNA filtering, and annotation of the mRNA
198	
199	Supplemental File 3 : Sequencing Batch Reactor Monod model for multiple-species
200	
201	
202	
203	

Supplemental File 1 : Phyloseq processing of the 16S rRNA reads

All commands detailed below were run using R v 3.5.1.

```
#Install required packages if not already available on the system
#=====
===
#source("https://bioconductor.org/biocLite.R")
#biocLite('DESeq2')
#biocLite('phyloseq')
#install.packages("ggfortify")
#install.packages("colorspace")
#install.packages("dplyr")
#install.packages("iNEXT")

#=====
===
#Load required libraries
#=====
===
library(phyloseq); packageVersion("phyloseq")      #At time of implementation, v 1.24.2
library(ggplot2); packageVersion("ggplot2")        #At time of implementation, v 2.2.1
library(reshape2); packageVersion("reshape2")      #At time of implementation, v 1.4.2
library(dplyr); packageVersion("dplyr")             #At time of implementation, v 0.7.1
library(grid); packageVersion("grid")               #At time of implementation, v 3.3.2
library(ggfortify); packageVersion("ggfortify")     #At time of implementation, v 0.4.1
library(gridExtra); packageVersion("gridExtra")     #At time of implementation, v 2.2.1
library(DESeq2); packageVersion("DESeq2")           #At time of implementation, v 1.14.1
library(biomformat); packageVersion("biomformat")   #At time of implementation, v 1.2.0
library(vegan)
library(micorbiomeSeq)

#=====
===
# Import Files, these files are provided in the Supplemental Data .zip package
#=====
===

otufilename_b1 <- "B1_ZOTU_Count_Sintax_SRT.txt"
mapfilename_b1 <- "B1_metafile_SRT.txt"
treefilename_b1 <- "B1_ZOTU_CLU.tre"
refseqfilename_b1 <- "B1_ZOTU.fa"

otufilename_b2 <- "B2_ZOTU_Count_Syntax_SRT.txt"
mapfilename_b2 <- "B2_metafile_SRT.txt"
treefilename_b2 <- "B2_ZOTU_CLU.tre"
refseqfilename_b2 <- "B2_ZOTU.fa"

B1 <- import_qiime(otufilename = otufilename_b1, mapfilename = mapfilename_b1, treefilename = treefilename_b1,
refseqfilename = refseqfilename_b1)
B2 <- import_qiime(otufilename = otufilename_b2, mapfilename = mapfilename_b2, treefilename = treefilename_b2,
refseqfilename = refseqfilename_b2)

Exp1_B1 <- subset_samples(B1, Exp == "Exp1")
Exp1_B2 <- subset_samples(B2, Exp == "Exp1")
Exp2_B1 <- subset_samples(B1, Exp == "Exp2")
Exp2_B2 <- subset_samples(B2, Exp == "Exp2")

write.table(sample_sums(B1),"sample_sumsB1.txt")
#=====
===
# Counts of reads per sample
#=====
===
sort(sample_sums(Exp1_B1))
sort(sample_sums(Exp1_B2))
sort(sample_sums(Exp2_B1))
sort(sample_sums(Exp2_B2))
```

```

271 #=====
272 ==
273 # Plot abundance per sample to check for any low abundance remaining samples
274 #=====
275 ==
276 sumsE1B1 <- data.frame(sample_sums(Exp1_B1))
277 sumsE1B1$sampleID <- rownames(sumsE1B1)
278 sumsE1B1$sampleID <- c("1d cDNA 1", "1d cDNA 2", "1d cDNA 3", "3d cDNA 1", "3d cDNA 2", "3d cDNA 3", "5d cDNA
279 1", "5d cDNA 2", "5d cDNA 3",
280 "7d cDNA 1", "7d cDNA 2", "7d cDNA 3", "10d cDNA 1", "10d cDNA 2", "10d cDNA 3", "15d
281 cDNA 1", "15d cDNA 2", "15d cDNA 3",
282 "1d gDNA 1", "1d gDNA 2", "1d gDNA 3", "3d gDNA 1", "3d gDNA 2", "3d gDNA 3", "5d gDNA
283 1", "5d gDNA 2", "5d gDNA 3",
284 "7d gDNA 1", "7d gDNA 2", "7d gDNA 3", "10d gDNA 1", "10d gDNA 2", "10d gDNA 3", "15d
285 gDNA 1", "15d gDNA 2", "15d gDNA 3",
286 "[+] gDNA 1", "[+] gDNA 2", "[+] gDNA 3")
287 sumsE1B1$sampleID <- factor(sumsE1B1$sampleID, levels = sumsE1B1$sampleID)
288 E1B1bars <- ggplot(sumsE1B1, aes(sampleID, sample_sums.Exp1_B1.)) +
289   geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
290 = 0.5)) +
291   labs(y = "ESV Count", title = "TP1 B1")
292
293 sumsE1B2 <- data.frame(sample_sums(Exp1_B2))
294 sumsE1B2$sampleID <- rownames(sumsE1B2)
295 sumsE1B2$sampleID <- sumsE1B1$sampleID
296 sumsE1B2$sampleID <- factor(sumsE1B2$sampleID, levels = sumsE1B2$sampleID)
297 E1B2bars <- ggplot(sumsE1B2[sumsE1B2$sampleID != "[+] gDNA 3",], aes(sampleID, sample_sums.Exp1_B2.)) +
298   geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
299 = 0.5)) +
300   labs(y = "ESV Count", title = "TP1 B2")
301
302 sumsE2B1 <- data.frame(sample_sums(Exp2_B1))
303 sumsE2B1$sampleID <- rownames(sumsE2B1)
304 sumsE2B1$sampleID <- c("1d cDNA 1", "1d cDNA 2", "1d cDNA 3", "3d cDNA 1", "3d cDNA 2", "3d cDNA 3", "5d cDNA
305 1", "5d cDNA 2", "5d cDNA 3",
306 "7d cDNA 1", "7d cDNA 2", "7d cDNA 3", "10d cDNA 1", "10d cDNA 2", "10d cDNA 3", "15d
307 cDNA 1", "15d cDNA 2", "15d cDNA 3",
308 "1d gDNA 1", "1d gDNA 2", "1d gDNA 3", "3d gDNA 1", "3d gDNA 2", "3d gDNA 3", "5d gDNA
309 1", "5d gDNA 2", "5d gDNA 3",
310 "7d gDNA 1", "7d gDNA 2", "7d gDNA 3", "10d gDNA 1", "10d gDNA 2", "10d gDNA 3", "15d
311 gDNA 1", "15d gDNA 2", "15d gDNA 3",
312 "[-] gDNA")
313 sumsE2B1$sampleID <- factor(sumsE2B1$sampleID, levels = sumsE2B1$sampleID)
314 E2B1bars <- ggplot(sumsE2B1, aes(sampleID, sample_sums.Exp2_B1.)) +
315   geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
316 = 0.5)) +
317   labs(y = "ESV Count", title = "TP2 B1")
318
319 sumsE2B2 <- data.frame(sample_sums(Exp2_B2))
320 sumsE2B2$sampleID <- rownames(sumsE2B2)
321 sumsE2B2$sampleID <- c("1d cDNA 1", "1d cDNA 2", "1d cDNA 3", "3d cDNA 1", "3d cDNA 2", "3d cDNA 3", "5d cDNA
322 1", "5d cDNA 2", "5d cDNA 3",
323 "7d cDNA 1", "7d cDNA 2", "7d cDNA 3", "10d cDNA 1", "10d cDNA 2", "10d cDNA 3", "15d
324 cDNA 1", "15d cDNA 2", "15d cDNA 3",
325 "1d gDNA 1", "1d gDNA 2", "1d gDNA 3", "3d gDNA 1", "3d gDNA 2", "3d gDNA 3", "5d gDNA
326 1", "5d gDNA 2", "5d gDNA 3",
327 "7d gDNA 1", "7d gDNA 2", "7d gDNA 3", "10d gDNA 1", "10d gDNA 2", "10d gDNA 3", "15d
328 gDNA 1", "15d gDNA 2", "15d gDNA 3",
329 "[-] cDNA 1", "[-] cDNA 2", "[-] cDNA 3", "[+] gDNA 1", "[+] gDNA 2", "[-] gDNA 1",
330 "[-] gDNA 2")
331 sumsE2B2$sampleID <- factor(sumsE2B2$sampleID, levels = sumsE2B2$sampleID)
332 E2B2bars <- ggplot(sumsE2B2, aes(sampleID, sample_sums.Exp2_B2.)) +
333   geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
334 = 0.5)) +
335   labs(y = "ESV Count", title = "TP2 B2")
336
337 #abun_bar_e1_b1 <- plot_bar(Exp1_B1, title = "Exp1 Primer B1")
338 #abun_bar_e1_b2 <- plot_bar(Exp1_B2, title = "Exp1 Primer B2")
339 #abun_bar_e2_b1 <- plot_bar(Exp2_B1, title = "Exp2 Primer B1")
340 #abun_bar_e2_b2 <- plot_bar(Exp2_B2, title = "Exp2 Primer B2")

```

```

341
342 pdf("Supplemental Abundance BarPlot.pdf", height = 8.6, width = 10.7)
343 grid.arrange(E1B1bars, E2B1bars, E1B2bars, E2B2bars, ncol = 2)
344 dev.off()
345
346 # => One of the positive controls B2pos_3_Exp1_B2 has 10x reads and should be removed
347 # => B1gR4_2_Exp2_B1 does not have enough reads and should be removed
348
349 Exp1_B2 <- subset_samples(Exp1_B2, X.SampleID != "B2pos_3_Exp1_B2")
350 Exp2_B1 <- subset_samples(Exp2_B1, X.SampleID != "B1gR4_2_Exp2_B1")
351
352 #=====
353 ===
354 #Plot rarefaction curve per sample
355 #=====
356 ===
357
358 colscale <-
359 c("gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blue",
360 "e","blue","black","black","black",
361
362 "gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blue"
363 ,"blue","black","black","black",
364 "grey","grey","grey")
365
366 colscale2 <-
367 c("gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blue",
368 "e","blue","black","black","black",
369 "gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","blue","blue","blue",
370 "black","black","black",
371 "grey","grey","grey")
372
373 ltyscale <- c(rep(1,18),rep(2,18),1,1,1)
374 ltyscale2 <- c(rep(1,18),rep(2,17),1,1,1)
375
376 pdf("Supplemental Rarefaction Curves.pdf", height = 8.6, width = 12.6)
377 par(mfrow=c(2,2),bty="n")
378 rarecurve(t(otu_table(Exp1_B1)), step=1000, lty=ltyscale, col=colscale, label=FALSE, xlab="16S Reads Exp1
379 Primer B1", ylab="ZOTU", ylim=c(0,4000), xlim=c(0,200000))
380 rarecurve(t(otu_table(Exp1_B2)), step=1000, lty=ltyscale, col=colscale, label=FALSE, xlab="16S Reads Exp1
381 Primer B2", ylab="ZOTU", ylim=c(0,4000), xlim=c(0,200000))
382 rarecurve(t(otu_table(Exp2_B1)), step=1000, lty=ltyscale2, col=colscale2, label=FALSE, xlab="16S Reads
383 Exp2 Primer B1", ylab="ZOTU", ylim=c(0,4000), xlim=c(0,200000))
384 rarecurve(t(otu_table(Exp2_B2)), step=1000, lty=c(ltyscale,1,1,1), col=c(colscale,rep("grey",6)), ylab="
385 ZOTU", label=FALSE, xlab="16S Reads Exp2 Primer B2", ylim=c(0,4000), xlim=c(0,200000))
386 dev.off()
387 dev.off()
388
389 #=====
390 ===
391 # Rarefy data for each B1 and B2 primer combinations seperately.
392 #=====
393 ===
394 Exp1_B1_even = rarefy_even_depth(subset_samples(Exp1_B1, SludgeType == "SRT"), rngseed = 1013)
395 Exp1_B2_even = rarefy_even_depth(subset_samples(Exp1_B2, SludgeType == "SRT"), rngseed = 1013)
396 Exp2_B1_even = rarefy_even_depth(subset_samples(Exp2_B1, SludgeType == "SRT"), rngseed = 1013)
397 Exp2_B2_even = rarefy_even_depth(subset_samples(Exp2_B2, SludgeType == "SRT"), rngseed = 1013)
398
399 sample_sums(Exp1_B1_even)
400 #=====
401 ===
402 #=====
403 ===
404 # Produce final summary graphic, run in text analyses
405 #=====
406 ===
407
408 # Set the colors of the top orders
409 #=====
410 order_colors = rep(c(

```

```

411     Bdellovibrionales='gray2',
412     Burkholderiales='gray30',
413     Caldilineales = 'indianred3',
414     Flavobacteriales='gray50',
415     Lactobacillales='red3',
416     Micrococcales = 'firebrick1',
417     Myxococcales='lightskyblue3',
418     Nitrosomonadales='gold1',
419     Planctomycetales='rosybrown1',
420     Pseudomonadales='dodgerblue4',
421     Rhodobacterales='royalblue1',
422     Rhodocyclales='orange3',
423     Sphingobacteriales='lightgoldenrod2',
424     Thiotrichales='darkorange4',
425     Verrucomicrobiales='navajowhite4'
426 ), 20)
427 #=====
428 # Aggregate samples into orders, plot abundance bars
429 #=====
430 glomExp1_B1_even_c <- tax_glom(subset_samples(Exp1_B1_even, Type %in% c("16S rRNA","16S rDNA")),
431 taxrank='Order')
432 div_by_rep_for_mean_rollup <- otu_table(glomExp1_B1_even_c)
433 otu_table(glomExp1_B1_even_c) <- div_by_rep_for_mean_rollup/c(rep(3,36))
434 gE1B1ec_filt = filter_taxa(glomExp1_B1_even_c, function(x) max(x) > 0, TRUE)
435 glomExp1_B1_even_c.long <- psmelt(gE1B1ec_filt)
436 glomExp1_B1_even_c.long <- arrange(glomExp1_B1_even_c.long, Order)
437
438 write.table(glomExp1_B1_even_c.long,"longcheck.txt")
439
440 bar_e1_b1 <- ggplot(glomExp1_B1_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
441 facet_grid(Type~.) +
442   geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
443 ggtitle("c. Exp1 Taxonomic Orders") +
444   scale_fill_manual(values = order_colors, na.value = 'white') +
445   theme(legend.position="none") +
446   scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" = "7","SRT5" = "10","SRT6" =
447 "15")) +
448   theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
449 element_blank(), strip.background = element_blank(),
450         axis.line = element_blank(), panel.grid.major = element_line(colour="white"),
451 panel.grid.minor = element_line(colour="white"),
452         panel.background = element_blank(), panel.border = element_blank())
453
454 orders10_exp1 <- names(sort(taxa_sums(subset_samples(glomExp1_B1_even_c, Type == "16S rRNA")), TRUE))
455 taxon10_exp1 <- tax_table(glomExp1_B1_even_c)
456 otu10_exp1 <- otu_table(glomExp1_B1_even_c)
457 write.table(cbind(otu10_exp1,taxon10_exp1),"Investigationofexp1rRNAotu.txt")
458 genus_exp1 <-taxon10_exp1 [orders10_exp1 ]
459
460 glomExp1_B2_even_c <- tax_glom(subset_samples(Exp1_B2_even, Type %in% c("16S rRNA","16S rDNA")),
461 taxrank='Order')
462 div_by_rep_for_mean_rollup <- otu_table(glomExp1_B2_even_c)
463 otu_table(glomExp1_B2_even_c) <- div_by_rep_for_mean_rollup/c(rep(3,36))
464 gE1B2ec_filt = filter_taxa(glomExp1_B2_even_c, function(x) max(x) > 0, TRUE)
465 glomExp1_B2_even_c.long <- psmelt(gE1B2ec_filt)
466 glomExp1_B2_even_c.long <- arrange(glomExp1_B2_even_c.long, Order)
467
468 orders10_exp1 <- names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])
469 sample_sums(glomExp1_B2_even_c) #83962
470 taxon10_exp1 <- tax_table(glomExp1_B2_even_c)
471 genus_exp1 <-taxon10_exp1 [orders10_exp1 ]
472 otu_count_exp1_RNA <- otu_table(subset_samples(glomExp1_B2_even_c, Type == "16S rRNA"))
473 otu_count_exp1_DNA <- otu_table(subset_samples(glomExp1_B2_even_c, Type == "16S rDNA"))
474
475 RNA_orders_exp1 <- otu_count_exp1_RNA[names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])]
476 DNA_orders_exp1 <- otu_count_exp1_DNA[names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])]
477
478 cor(RNA_orders_exp1,DNA_orders_exp1, method = "spearman")
479

```



```

480 bar_e1_b2 <- ggplot(glomExp1_B2_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
481 facet_grid(Type~.) +
482   geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
483 ggtitle("c. Exp1 Taxonomic Orders") +
484   scale_fill_manual(values = order_colors, na.value = 'white') +
485   theme(legend.position="none") +
486   scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" = "7", "SRT5" = "10", "SRT6" =
487 "15")) +
488   theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
489 element_blank(), strip.background = element_blank(),
490   axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
491 element_line(colour="white"),
492   panel.background = element_blank(), panel.border = element_blank())
493
494 glomExp2_B1_even_c <- tax_glom(subset_samples(Exp2_B1_even, Type %in% c("16S rRNA", "16S rDNA")),
495 taxrank='Order')
496 div_by_rep_for_mean_rollup <- otu_table(glomExp2_B1_even_c)
497 div_by_rep_for_mean_rollup[,28:29] <- div_by_rep_for_mean_rollup[,28:29]*1.5
498 otu_table(glomExp2_B1_even_c) <- div_by_rep_for_mean_rollup/3
499 gE2B1ec_filt = filter_taxa(glomExp2_B1_even_c, function(x) max(x) > 0, TRUE)
500 glomExp2_B1_even_c.long <- psmelt(gE2B1ec_filt)
501 glomExp2_B1_even_c.long <- arrange(glomExp2_B1_even_c.long, Order)
502
503 bar_e2_b1 <- ggplot(glomExp2_B1_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
504 facet_grid(Type~.) +
505   geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
506 ggtitle("d. Exp 2") +
507   scale_fill_manual(values = order_colors, na.value = "white") + theme(legend.position="none") +
508   scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" = "7", "SRT5" = "10", "SRT6" =
509 "15")) +
510   theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
511 element_blank(), strip.background = element_blank(),
512   axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
513 element_line(colour="white"),
514   panel.background = element_blank(), panel.border = element_blank())
515
516 glomExp2_B2_even_c <- tax_glom(subset_samples(Exp2_B2_even, Type %in% c("16S rRNA", "16S rDNA")),
517 taxrank='Order')
518 div_by_rep_for_mean_rollup <- otu_table(glomExp2_B2_even_c)
519 otu_table(glomExp2_B2_even_c) <- div_by_rep_for_mean_rollup/3
520 gE2B2ec_filt = filter_taxa(glomExp2_B2_even_c, function(x) max(x) > 0, TRUE)
521 glomExp2_B2_even_c.long <- psmelt(gE2B2ec_filt)
522 glomExp2_B2_even_c.long <- arrange(glomExp2_B2_even_c.long, Order)
523
524 orders10_exp2 <- names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:20])
525 sample_sums(glomExp2_B2_even_c) #83962
526 taxon10_exp2 <- tax_table(glomExp2_B2_even_c)
527 genus_exp2 <- taxon10_exp2[orders10_exp2]
528 otu_count_exp2 <- otu_table(glomExp2_B2_even_c)
529
530 otu_count_exp2_RNA <- otu_table(subset_samples(glomExp2_B2_even_c, Type == "16S rRNA"))
531 otu_count_exp2_DNA <- otu_table(subset_samples(glomExp2_B2_even_c, Type == "16S rDNA"))
532
533 RNA_orders_exp2 <- otu_count_exp2_RNA[names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:50])]
534 DNA_orders_exp2 <- otu_count_exp2_DNA[names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:50])]
535
536 write.table(cor(RNA_orders_exp2, DNA_orders_exp2, method = "spearman"), "exp2ordercorrelation.txt")
537 write.table(cor(RNA_orders_exp1, DNA_orders_exp2, method = "spearman"), "exp1ordercorrelation.txt")
538
539
540 bar_e2_b2 <- ggplot(glomExp2_B2_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
541 facet_grid(Type~.) +
542   geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
543 ggtitle("d. Exp 2") +
544   scale_fill_manual(values = order_colors, na.value = "white") + theme(legend.position="none") +
545   scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" = "7", "SRT5" = "10", "SRT6" =
546 "15")) +
547   theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
548 element_blank(), strip.background = element_blank(),

```

```

549     axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
550     element_line(colour="white"),
551     panel.background = element_blank(), panel.border = element_blank())
552
553 #=====
554 # In text analysis: Comparison of 15d to 1d for Burkholderiales, Rhodocyclales, and Myxococcales
555 #=====
556 burk_count_exp1 <- otu_count_exp1 ["ZOTU28",]/83962*100
557 burk_analysis_exp1 <-
558 c(mean(burk_count_exp1[1,1:3]),sd(burk_count_exp1[1,1:3]),mean(burk_count_exp1[1,16:18]),sd(burk_count_exp
559 1[1,16:18]))
560
561 rhodo_count_exp1 <- otu_count_exp1["ZOTU2",]/54554*100
562 rhodo_analysis_exp1 <-
563 c(mean(rhodo_count_exp1[1,1:3]),sd(rhodo_count_exp1[1,1:3]),mean(rhodo_count_exp1[1,16:18]),sd(rhodo_count
564 _exp1[1,16:18]))
565
566 myxo_count_exp1 <- otu_count_exp1["ZOTU284",]/54554*100
567 myxo_analysis_exp1 <-
568 c(mean(myxo_count_exp1[1,1:3]),sd(myxo_count_exp1[1,1:3]),mean(myxo_count_exp1[1,16:18]),sd(myxo_count_exp
569 1[1,16:18]))
570
571 burk_count_exp2 <- otu_count_exp2["ZOTU10",]/54554*100
572 burk_analysis_exp2 <-
573 c(mean(burk_count_exp2[1,1:3]),sd(burk_count_exp2[1,1:3]),mean(burk_count_exp2[1,16:18]),sd(burk_count_exp
574 2[1,16:18]))
575
576 rhodo_count_exp2 <- otu_count_exp2["ZOTU2",]/54554*100
577 rhodo_analysis_exp2 <-
578 c(mean(rhodo_count_exp2[1,1:3]),sd(rhodo_count_exp2[1,1:3]),mean(rhodo_count_exp2[1,16:18]),sd(rhodo_count
579 _exp2[1,16:18]))
580
581 myxo_count_exp2 <- otu_count_exp2["ZOTU20",]/54554*100
582 myxo_analysis_exp2 <-
583 c(mean(myxo_count_exp2[1,1:3]),sd(myxo_count_exp2[1,1:3]),mean(myxo_count_exp2[1,16:18]),sd(myxo_count_exp
584 2[1,16:18]))
585
586 #=====
587 # Create appropriate labels as grobs.
588 #=====
589
590 planc <- textGrob("Planctomycetales",gp=gpar(col="rosybrown1", fontsize=10,
591 fontface="bold.italic"),x=0.1,just="left")
592 cald <- textGrob("Caldilineales",gp=gpar(col="indianred3", fontsize=10,
593 fontface="bold.italic"),x=0.1,just="left")
594 micro <- textGrob("Micrococcales",gp=gpar(col="firebrick1", fontsize=10,
595 fontface="bold.italic"),x=0.1,just="left")
596 bdel <- textGrob("Bdellovibrionales",gp=gpar(col="grey2", fontsize=10,
597 fontface="bold.italic"),x=0.1,just="left")
598 burk <- textGrob("Burkholderiales",gp=gpar(col="gray30", fontsize=10,
599 fontface="bold.italic"),x=0.1,just="left")
600 flav <- textGrob("Flavobacteriales",gp=gpar(col="gray50", fontsize=10,
601 fontface="bold.italic"),x=0.1,just="left")
602 myx <- textGrob("Myxococcales",gp=gpar(col="lightskyblue3", fontsize=10,
603 fontface="bold.italic"),x=0.1,just="left")
604 nitro <- textGrob("Nitrosomonadales",gp=gpar(col="gold1", fontsize=10,
605 fontface="bold.italic"),x=0.1,just="left")
606 lacto <- textGrob("Lactobacillales",gp=gpar(col="red3", fontsize=10,
607 fontface="bold.italic"),x=0.1,just="left")
608 sphingo <- textGrob("Sphingobacteriales",gp=gpar(col="lightgoldenrod2", fontsize=10,
609 fontface="bold.italic"),x=0.1,just="left")
610 rhodo <- textGrob("Rhodocyclales",gp=gpar(col="orange3", fontsize=10,
611 fontface="bold.italic"),x=0.1,just="left")
612 rhodob <- textGrob("Rhodobacterales",gp=gpar(col="royalblue1", fontsize=10,
613 fontface="bold.italic"),x=0.1,just="left")
614 psd <- textGrob("Pseudomonadales",gp=gpar(col="dodgerblue4", fontsize=10,
615 fontface="bold.italic"),x=0.1,just="left")
616 thio <- textGrob("Thiotrichales",gp=gpar(col="darkorange4", fontsize=10,
617 fontface="bold.italic"),x=0.1,just="left")

```

```

618 ver <- textGrob("Verrucomicrobiales",gp=gpar(col="navajowhite4", fontsize=10,
619 fontface="bold.italic"),x=0.1,just="left")
620 all_else <- textGrob("Other",gp=gpar(col="white", fontsize=10, fontface="bold.italic"),x=0.1,just="left")
621
622 theme_set(theme_bw())
623
624 blank <- textGrob("",gp=gpar(col="black"))
625 leg_cDNA <- textGrob(". 16S rRNA",gp=gpar(col="black", fontsize=10, fontface="bold"),x=0.1,just="left")
626 leg_gDNA <- textGrob(". 16S rDNA",gp=gpar(col="light blue", fontsize=10,
627 fontface="bold"),x=0.1,just="left")
628 title_exp1 <- textGrob("Exp1 48d",gp=gpar(col="black", fontsize=16, fontface="bold"),x=0.1,just="left")
629 title_exp2 <- textGrob("Exp2 187d",gp=gpar(col="black", fontsize=16, fontface="bold"),x=0.1,just="left")
630 xaxis_lab <- textGrob("MRT [d]",gp=gpar(col="black", fontsize=12, fontface="bold"),x=0.1,just="left")
631
632 #=====
633 # Plot the richness versus MRT
634 #=====
635 Exp1_B1_richness_obs <- plot_richness(Exp1_B1_even, "ST_RN", "Type", title = "a. Richness",
636 measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
637 theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" =
638 "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
639 element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
640 c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
641 panel.border = element_blank(), panel.background = element_blank())
642
643 Exp1_B2_richness_obs <- plot_richness(Exp1_B2_even, "ST_RN", "Type", title = "a. Richness",
644 measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
645 theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" =
646 "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
647 element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
648 c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
649 panel.border = element_blank(), panel.background = element_blank())
650
651 Exp2_B1_richness_obs <- plot_richness(Exp2_B1_even, "ST_RN", "Type", title = "b. Richness",
652 measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
653 theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" =
654 "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
655 element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
656 c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
657 panel.border = element_blank(), panel.background = element_blank())
658
659 Exp2_B2_richness_obs <- plot_richness(Exp2_B2_even, "ST_RN", "Type", title = "b. Richness",
660 measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
661 theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" = "5", "SRT4" =
662 "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
663 element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
664 c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
665 panel.border = element_blank(), panel.background = element_blank())
666
667 rich <- rbind(Exp1_B2_richness_obs$data, Exp2_B2_richness_obs$data)
668 write.table(rich, "ASVrich.txt")
669
670 #=====
671 # Plot the Shannon Diversity vs MRT
672 # Note : must consider raw data.
673 #=====
674 Exp1_B1_richness_sha <- plot_richness(subset_samples(Exp1_B1, SludgeType == "SRT"), "ST_RN", "Type",
675 title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
676 "black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" =
677 "5", "SRT4" = "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
678 axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
679 breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
680 panel.border = element_blank(), panel.background = element_blank())
681 Exp1_B2_richness_sha <- plot_richness(subset_samples(Exp1_B2, SludgeType == "SRT"), "ST_RN", "Type",
682 title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
683 "black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" =
684 "5", "SRT4" = "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
685 axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
686 breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(), strip.text.x = element_blank(),
687 panel.border = element_blank(), panel.background = element_blank())

```

```

688 Exp2_B1_richness_sha <- plot_richness(subset_samples(Exp2_B1, SludgeType == "SRT"), "ST_RN", "Type",
689 title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
690 "black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" =
691 "5", "SRT4" = "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
692 axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
693 breaks = c(2, 3, 4, 5, 6)) + theme(strip.background = element_blank(), strip.text.x = element_blank(),
694 panel.border = element_blank(), panel.background = element_blank())
695 Exp2_B2_richness_sha <- plot_richness(subset_samples(Exp2_B2_even, SludgeType == "SRT"), "ST_RN", "Type",
696 title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
697 "black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3", "SRT3" =
698 "5", "SRT4" = "7", "SRT5" = "10", "SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
699 axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
700 breaks = c(2, 3, 4, 5, 6)) + theme(strip.background = element_blank(), strip.text.x = element_blank(),
701 panel.border = element_blank(), panel.background = element_blank())
702
703 #=====
704 # Run all correlation comparisons
705 #=====
706
707 shan <- rbind(Exp1_B2_richness_sha$data, Exp2_B2_richness_sha$data)
708 write.table(shan, "ASVshan.txt")
709 diversity_sum <- data.frame(rich$X.SampleID, rich$Type, rich$Exp, rich$ReactorNum, rich$value, shan$value)
710
711 exp1_rRNA_div <- diversity_sum[which(diversity_sum$rich.Type=="16S rRNA" & diversity_sum$rich.Exp ==
712 "Exp1"),]
713 exp2_rRNA_div <- diversity_sum[which(diversity_sum$rich.Type=="16S rRNA" & diversity_sum$rich.Exp ==
714 "Exp2"),]
715 exp1_rDNA_div <- diversity_sum[which(diversity_sum$rich.Type=="16S rDNA" & diversity_sum$rich.Exp ==
716 "Exp1"),]
717 exp2_rDNA_div <- diversity_sum[which(diversity_sum$rich.Type=="16S rDNA" & diversity_sum$rich.Exp ==
718 "Exp2"),]
719
720 e1_rich_p <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp1_rDNA_div,
721 tapply(rich.value, rich.ReactorNum, mean)))
722 e1_rich_s <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp1_rDNA_div,
723 tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
724 e1_rich_l <- sqrt(summary(lm(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum,
725 mean))~with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
726 e1_shan_p <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)), with(exp1_rDNA_div,
727 tapply(shan.value, rich.ReactorNum, mean)))
728 e1_shan_s <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)), with(exp1_rDNA_div,
729 tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
730 e1_shan_l <- summary(lm(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp1_rDNA_div,
731 tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared
732
733 e2_rich_p <- cor(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp2_rDNA_div,
734 tapply(rich.value, rich.ReactorNum, mean)))
735 e2_rich_s <- cor(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp2_rDNA_div,
736 tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
737 e2_rich_l <- sqrt(summary(lm(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum,
738 mean))~with(exp2_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
739 e2_shan_p <- cor(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)), with(exp2_rDNA_div,
740 tapply(shan.value, rich.ReactorNum, mean)))
741 e2_shan_s <- cor(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)), with(exp2_rDNA_div,
742 tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
743 e2_shan_l <- summary(lm(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rDNA_div,
744 tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared
745
746 DNA_RNA_pearson_summary <- data.frame(e1_rich_p, e1_shan_p, e2_rich_p, e2_shan_p)
747 DNA_RNA_spearman_summary <- data.frame(e1_rich_s, e1_shan_s, e2_rich_s, e2_shan_s)
748
749 e1_rich_p <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp2_rRNA_div,
750 tapply(rich.value, rich.ReactorNum, mean)))
751 e1_rich_s <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)), with(exp2_rRNA_div,
752 tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
753 e1_rich_l <- sqrt(summary(lm(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum,
754 mean))~with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
755 e1_shan_p <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)), with(exp2_rRNA_div,
756 tapply(shan.value, rich.ReactorNum, mean)))

```

```

757 e1_shan_s <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rRNA_div,
758 tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
759 e1_shan_l <- summary(lm(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rRNA_div,
760 tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared
761
762 e1_rich_p <- cor(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
763 tapply(rich.value, rich.ReactorNum, mean)))
764 e1_rich_s <- cor(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
765 tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
766 e1_rich_l <- sqrt(summary(lm(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum,
767 mean))~with(exp2_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
768 e1_shan_p <- cor(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
769 tapply(shan.value, rich.ReactorNum, mean)))
770 e1_shan_s <- cor(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
771 tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
772 e1_shan_l <- summary(lm(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rDNA_div,
773 tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared
774
775 DNA_RNA_pearson_summary <- data.frame(e1_rich_p,e1_shan_p, e2_rich_p, e2_shan_p)
776 DNA_RNA_spearman_summary <- data.frame(e1_rich_s,e1_shan_s, e2_rich_s, e2_shan_s)
777
778 MRTe1_rich_p <- cor(exp1_rRNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum)
779 MRTe1_rich_s <- cor(exp1_rRNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
780 MRTe1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
781 MRTe1_shan_p <- cor(exp1_rRNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum)
782 MRTe1_shan_s <- cor(exp1_rRNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
783 MRTe1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
784
785 MRTe2_rich_p <- cor(exp2_rRNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum)
786 MRTe2_rich_s <- cor(exp2_rRNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
787 MRTe2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
788 MRTe2_shan_p <- cor(exp2_rRNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum)
789 MRTe2_shan_s <- cor(exp2_rRNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
790 MRTe2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
791
792 MRT_rRNA_pearson_summary <- data.frame(MRTe1_rich_p,MRTe1_shan_p, MRTe2_rich_p, MRTe2_shan_p)
793 MRT_rRNA_spearman_summary <- data.frame(MRTe1_rich_s, MRTe1_shan_s, MRTe2_rich_s, MRTe2_shan_s)
794
795 MRTe1_rich_p <- cor(exp1_rDNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum)
796 MRTe1_rich_s <- cor(exp1_rDNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
797 MRTe1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
798 MRTe1_shan_p <- cor(exp1_rDNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum)
799 MRTe1_shan_s <- cor(exp1_rDNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
800 MRTe1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
801
802 MRTe2_rich_p <- cor(exp2_rDNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum)
803 MRTe2_rich_s <- cor(exp2_rDNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
804 MRTe2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
805 MRTe2_shan_p <- cor(exp2_rDNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum)
806 MRTe2_shan_s <- cor(exp2_rDNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
807 MRTe2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
808
809 MRT_rDNA_pearson_summary <- data.frame(MRTe1_rich_p,MRTe1_shan_p, MRTe2_rich_p, MRTe2_shan_p)
810 MRT_rDNA_spearman_summary <- data.frame(MRTe1_rich_s, MRTe1_shan_s, MRTe2_rich_s, MRTe2_shan_s)
811
812 mRNA_exp1_rich <- rep(c(2949.8, 2968.3, 3068.4, 3214.5, 3348.0, 3362.7), each=3)
813 mRNA_exp2_rich <- rep(c(3001.3, 2994.5, 3034.3, 3322.0, 3382.9, 3350.3), each=3)
814 mRNA_exp1_shan <- rep(c(5.9916,6.01818,6.08388,6.06815,5.9818,6.00447), each=3)
815 mRNA_exp2_shan <- rep(c(6.00442,6.01225,6.08577,6.05381,6.01788,6.0194), each=3)
816
817 rDNA_mRNA_e1_rich_p <- cor(exp1_rDNA_div$rich.value,mRNA_exp1_rich)
818 rDNA_mRNA_e1_rich_s <- cor(exp1_rDNA_div$rich.value,mRNA_exp1_rich, method = "spearman")
819 rDNA_mRNA_e1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~mRNA_exp1_rich))$adj.r.squared)
820
821 rRNA_mRNA_e1_rich_p <- cor(exp1_rRNA_div$rich.value,mRNA_exp1_rich)
822 rRNA_mRNA_e1_rich_s <- cor(exp1_rRNA_div$rich.value,mRNA_exp1_rich, method = "spearman")
823 rRNA_mRNA_e1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~mRNA_exp1_rich))$adj.r.squared)
824
825 rDNA_mRNA_e2_rich_p <- cor(exp2_rDNA_div$rich.value,mRNA_exp2_rich)
826 rDNA_mRNA_e2_rich_s <- cor(exp2_rDNA_div$rich.value,mRNA_exp2_rich, method = "spearman")

```

```

827 rDNA_mRNA_e2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~mRNA_exp2_rich))$adj.r.squared)
828
829 rRNA_mRNA_e2_rich_p <- cor(exp2_rRNA_div$rich.value,mRNA_exp2_rich)
830 rRNA_mRNA_e2_rich_s <- cor(exp2_rRNA_div$rich.value,mRNA_exp2_rich, method = "spearman")
831 rRNA_mRNA_e2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~mRNA_exp2_rich))$adj.r.squared)
832
833 rDNA_mRNA_e1_shan_p <- cor(exp1_rDNA_div$shan.value,mRNA_exp1_shan)
834 rDNA_mRNA_e1_shan_s <- cor(exp1_rDNA_div$shan.value,mRNA_exp1_shan, method = "spearman")
835 rDNA_mRNA_e1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~mRNA_exp1_shan))$adj.r.squared)
836
837 rRNA_mRNA_e1_shan_p <- cor(exp1_rRNA_div$shan.value,mRNA_exp1_shan)
838 rRNA_mRNA_e1_shan_s <- cor(exp1_rRNA_div$shan.value,mRNA_exp1_shan, method = "spearman")
839 rRNA_mRNA_e1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~mRNA_exp1_shan))$adj.r.squared)
840
841 rDNA_mRNA_e2_shan_p <- cor(exp2_rDNA_div$shan.value,mRNA_exp2_shan)
842 rDNA_mRNA_e2_shan_s <- cor(exp2_rDNA_div$shan.value,mRNA_exp2_shan, method = "spearman")
843 rDNA_mRNA_e2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~mRNA_exp2_shan))$adj.r.squared)
844
845 rRNA_mRNA_e2_shan_p <- cor(exp2_rRNA_div$shan.value,mRNA_exp2_shan)
846 rRNA_mRNA_e2_shan_s <- cor(exp2_rRNA_div$shan.value,mRNA_exp2_shan, method = "spearman")
847 rRNA_mRNA_e2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~mRNA_exp2_shan))$adj.r.squared)
848
849 model_rich <- rep(c(0.261844334, 0.318322406, 0.384583649, 0.435570161, 0.501653578, 0.579202516),each=3)
850
851 rDNA_model_e1_rich_p <- cor(exp1_rDNA_div$rich.value,model_rich)
852 rDNA_model_e1_rich_s <- cor(exp1_rDNA_div$rich.value,model_rich, method = "spearman")
853 rDNA_model_e1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~model_rich))$adj.r.squared)
854
855 rRNA_model_e1_rich_p <- cor(exp1_rRNA_div$rich.value,model_rich)
856 rRNA_model_e1_rich_s <- cor(exp1_rRNA_div$rich.value,model_rich, method = "spearman")
857 rRNA_model_e1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~model_rich))$adj.r.squared)
858
859 rDNA_model_e2_rich_p <- cor(exp2_rDNA_div$rich.value,model_rich)
860 rDNA_model_e2_rich_s <- cor(exp2_rDNA_div$rich.value,model_rich, method = "spearman")
861 rDNA_model_e2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~model_rich))$adj.r.squared)
862
863 rRNA_model_e2_rich_p <- cor(exp2_rRNA_div$rich.value,model_rich)
864 rRNA_model_e2_rich_s <- cor(exp2_rRNA_div$rich.value,model_rich, method = "spearman")
865 rRNA_model_e2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~model_rich))$adj.r.squared)
866
867 model_shan <- rep(c(2.56, 2.76, 2.90, 3.02, 3.11, 3.20),each=3)
868
869 rDNA_model_e1_shan_p <- cor(exp1_rDNA_div$shan.value,model_shan)
870 rDNA_model_e1_shan_s <- cor(exp1_rDNA_div$shan.value,model_shan, method = "spearman")
871 rDNA_model_e1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~model_shan))$adj.r.squared)
872
873 rRNA_model_e1_shan_p <- cor(exp1_rRNA_div$shan.value,model_shan)
874 rRNA_model_e1_shan_s <- cor(exp1_rRNA_div$shan.value,model_shan, method = "spearman")
875 rRNA_model_e1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~model_shan))$adj.r.squared)
876
877 rDNA_model_e2_shan_p <- cor(exp2_rDNA_div$shan.value,model_shan)
878 rDNA_model_e2_shan_s <- cor(exp2_rDNA_div$shan.value,model_shan, method = "spearman")
879 rDNA_model_e2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~model_shan))$adj.r.squared)
880
881 rRNA_model_e2_shan_p <- cor(exp2_rRNA_div$shan.value,model_shan)
882 rRNA_model_e2_shan_s <- cor(exp2_rRNA_div$shan.value,model_shan, method = "spearman")
883 rRNA_model_e2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~model_shan))$adj.r.squared)
884
885 #=====
886 # Establish layout grid
887 #=====
888 lay <- rbind(c(1,1,1,1,2,2,2,2,7),
889             c(24,24,25,25,26,26,27,27,7),
890             c(24,24,25,25,26,26,27,27,7),
891             c(24,24,25,25,26,26,27,27,7),
892             c(24,24,25,25,26,26,27,27,7),
893             c(24,24,25,25,26,26,27,27,5),
894             c(24,24,25,25,26,26,27,27,6),
895             c(24,24,25,25,26,26,27,27,8),
896             c(24,24,25,25,26,26,27,27,8),

```

```

897         c(24,24,25,25,26,26,27,27,8),
898         c(24,24,25,25,26,26,27,27,8),
899         c(24,24,25,25,26,26,27,27,8),
900         c(28,28,29,30,30,30,31,32,23),
901         c(32,32,32,32,32,32,32,32,23),
902         c(3,3,3,3,4,4,4,4,23),
903         c(3,3,3,3,4,4,4,4,23),
904         c(3,3,3,3,4,4,4,4,23),
905         c(3,3,3,3,4,4,4,4,23),
906         c(3,3,3,3,4,4,4,4,10),
907         c(3,3,3,3,4,4,4,4,11),
908         c(3,3,3,3,4,4,4,4,12),
909         c(3,3,3,3,4,4,4,4,13),
910         c(3,3,3,3,4,4,4,4,14),
911         c(3,3,3,3,4,4,4,4,19),
912         c(3,3,3,3,4,4,4,4,17),
913         c(3,3,3,3,4,4,4,4,18),
914         c(3,3,3,3,4,4,4,4,16),
915         c(3,3,3,3,4,4,4,4,20),
916         c(3,3,3,3,4,4,4,4,21),
917         c(3,3,3,3,4,4,4,4,22),
918         c(3,3,3,3,4,4,4,4,39),
919         c(3,3,3,3,4,4,4,4,15),
920         c(3,3,3,3,4,4,4,4,40),
921         c(3,3,3,3,4,4,4,4,9),
922         c(3,3,3,3,4,4,4,4,33),
923         c(3,3,3,3,4,4,4,4,33),
924         c(3,3,3,3,4,4,4,4,33),
925         c(34,34,35,36,36,36,37,38,33))
926
927 #=====
928 # Plot for primer B2
929 #=====
930 pdf("Figure 2.pdf", height = 7, width = 12)
931 grid.arrange(title_exp1, title_exp2, bar_e1_b2, bar_e2_b2, leg_cDNA,
932             leg_gDNA, blank, blank, planc, bdel,
933             burk, flav, myx, nitro, lacto,
934             sphingo, rhodob, rhodo, psd, thio,
935             ver, all_else, blank, Exp1_B2_richness_obs, Exp1_B2_richness_sha, Exp2_B2_richness_obs,
936             Exp2_B2_richness_sha, blank,
937             xaxis_lab, blank, xaxis_lab, blank, blank, blank, xaxis_lab, blank, xaxis_lab, blank, cald,
938             micro,
939             ncol=2, layout_matrix = lay)
940 dev.off()
941
942 #=====
943 # Plot for primer B1
944 #=====
945 pdf("Supplemental 16S Figure.pdf", height = 7, width = 12)
946 grid.arrange(title_exp1, title_exp2, bar_e1_b1, bar_e2_b1, leg_cDNA,
947             leg_gDNA, blank, blank, planc, bdel,
948             burk, flav, myx, nitro, lacto,
949             sphingo, rhodob, rhodo, psd, thio,
950             ver, all_else, blank, Exp1_B2_richness_obs, Exp1_B2_richness_sha, Exp2_B2_richness_obs,
951             Exp2_B2_richness_sha, blank,
952             xaxis_lab, blank, xaxis_lab, blank, blank, blank, xaxis_lab, blank, xaxis_lab, blank, cald,
953             micro,
954             ncol=2, layout_matrix = lay)
955 dev.off()
956
957 #=====
958 ===
959
960
961
962

```

Supplemental File 2 : In silico quality filtering, rRNA filtering, and annotation of the mRNA

All commands detailed here are able to be run on a Linux operating system with an Ubuntu distribution.

The first step in the analysis pipeline is to check the quality of the raw datafiles using the FastQC program:

```
fastqc *fastq.gz
```

Thereafter, trimmomatic is used to clip the adapter sequences within reads and remove sequences of low quality:

```
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67632_H5WH2BGX3_1_exp1R1_rna_epidemiology__ATCACG_S1_R1_001_MM_1.fastq.gz
Exp1_R1_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R1_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67633_H5WH2BGX3_1_exp1R2_rna_epidemiology__CGATGT_S2_R1_001_MM_1.fastq.gz
Exp1_R2_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R2_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67634_H5WH2BGX3_1_exp1R3_rna_epidemiology__TTAGGC_S3_R1_001_MM_1.fastq.gz
Exp1_R3_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R3_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67635_H5WH2BGX3_1_exp1R4_rna_epidemiology__TGACCA_S4_R1_001_MM_1.fastq.gz
Exp1_R4_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R4_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67636_H5WH2BGX3_1_exp1R5_rna_epidemiology__ACAGTG_S5_R1_001_MM_1.fastq.gz
Exp1_R5_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R5_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67637_H5WH2BGX3_1_exp1R6_rna_epidemiology__GCCAAT_S6_R1_001_MM_1.fastq.gz
Exp1_R6_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R6_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67638_H5WH2BGX3_1_exp2R1_rna_epidemiology__CAGATC_S7_R1_001_MM_1.fastq.gz
Exp2_R1_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R1_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67639_H5WH2BGX3_1_exp2R2_rna_epidemiology__ACTTGA_S8_R1_001_MM_1.fastq.gz
```



```

1014 Exp2_R2_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
1015 MINLEN:36 >& Exp2_R2_trim_log.txt &
1016 nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
1017 SE -threads 4
1018 BSSE_QGF_67640_H5WH2BGX3_1_exp2R3_rna_epidemiology__GATCAG_S9_R1_001_MM_1.fastq.gz
1019 Exp2_R3_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
1020 MINLEN:36 >& Exp2_R3_trim_log.txt &
1021 nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
1022 SE -threads 4
1023 BSSE_QGF_67641_H5WH2BGX3_1_exp2R4_rna_epidemiology__TAGCTT_S10_R1_001_MM_1.fastq.gz
1024 Exp2_R4_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
1025 MINLEN:36 >& Exp2_R4_trim_log.txt &
1026 nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
1027 SE -threads 4
1028 BSSE_QGF_67642_H5WH2BGX3_1_exp2R5_rna_epidemiology__GGCTAC_S11_R1_001_MM_1.fastq.gz
1029 Exp2_R5_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
1030 MINLEN:36 >& Exp2_R5_trim_log.txt &
1031 nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
1032 SE -threads 4
1033 BSSE_QGF_67643_H5WH2BGX3_1_exp2R6_rna_epidemiology__CTTGTA_S12_R1_001_MM_1.fastq.gz
1034 Exp2_R6_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
1035 MINLEN:36 >& Exp2_R6_trim_log.txt &

```

1036
1037 The quality of the trimmed datafiles are again checked using the FastQC program:

```

1038
1039 fastqc *trimmed.fq.gz
1040

```

1041 After ensuring that the rRNA databases are indexed, SortMeRNA is run to remove remaining rRNA

1042 encoding reads (an example is provided below for Experiment 1 Reactor 1):

```

1043
1044 gzip -d -k Exp1_R1_trimmed.fq.gz
1045 time ./sortmerna --ref ./rRNA_databases/silva-bac-16s-id90.fasta,./index/silva-bac-
1046 16s-db:./rRNA_databases/silva-bac-23s-id98.fasta,./index/silva-bac-23s-
1047 db:./rRNA_databases/silva-arc-16s-id95.fasta,./index/silva-arc-16s-
1048 db:./rRNA_databases/silva-arc-23s-id98.fasta,./index/silva-arc-23s-
1049 db:./rRNA_databases/silva-euk-18s-id95.fasta,./index/silva-euk-18s-
1050 db:./rRNA_databases/silva-euk-28s-id98.fasta,./index/silva-euk-
1051 28s:./rRNA_databases/rfam-5s-database-id98.fasta,./index/rfam-5s-
1052 db:./rRNA_databases/rfam-5.8s-database-id98.fasta,./index/rfam-5.8s-db --reads
1053 Exp1_R1_trimmed.fq --sam --num_alignments 1 --fastx --aligned Exp1_R1rRNA --other
1054 E1_R1_s_mRNA --log -v -a 20
1055 rm Exp1_R1_trimmed.fq
1056 rm Exp1_R1rRNA.fq
1057 rm Exp1_R1rRNA.sam
1058

```

1059 The files are unzipped and converted from fastq to fasta format:

```

1060
1061 nohup gzip -k E1_R1_s_mRNA.fq >& log1.out &
1062 nohup gzip -k E1_R2_s_mRNA.fq >& log2.out &
1063 nohup gzip -k E1_R3_s_mRNA.fq >& log3.out &
1064 nohup gzip -k E1_R4_s_mRNA.fq >& log4.out &
1065 nohup gzip -k E1_R5_s_mRNA.fq >& log5.out &
1066 nohup gzip -k E1_R6_s_mRNA.fq >& log6.out &
1067
1068 nohup gzip -k E2_R1_s_mRNA.fq >& log7.out &

```

```

1069 nohup gzip -k E2_R2_s_mRNA.fq >& log8.out &
1070 nohup gzip -k E2_R3_s_mRNA.fq >& log9.out &
1071 nohup gzip -k E2_R4_s_mRNA.fq >& log10.out &
1072 nohup gzip -k E2_R5_s_mRNA.fq >& log11.out &
1073 nohup gzip -k E2_R6_s_mRNA.fq >& log12.out &
1074
1075 cat E1_R1_s_mRNA.fq | perl -e
1076 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1077 E1_R1_s_mRNA.fasta
1078 cat E1_R2_s_mRNA.fq | perl -e
1079 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1080 E1_R2_s_mRNA.fasta
1081 cat E1_R3_s_mRNA.fq | perl -e
1082 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1083 E1_R3_s_mRNA.fasta
1084 cat E1_R4_s_mRNA.fq | perl -e
1085 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1086 E1_R4_s_mRNA.fasta
1087 cat E1_R5_s_mRNA.fq | perl -e
1088 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1089 E1_R5_s_mRNA.fasta
1090 cat E1_R6_s_mRNA.fq | perl -e
1091 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1092 E1_R6_s_mRNA.fasta
1093
1094 cat E2_R1_s_mRNA.fq | perl -e
1095 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1096 E2_R1_s_mRNA.fasta
1097 cat E2_R2_s_mRNA.fq | perl -e
1098 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1099 E2_R2_s_mRNA.fasta
1100 cat E2_R3_s_mRNA.fq | perl -e
1101 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1102 E2_R3_s_mRNA.fasta
1103 cat E2_R4_s_mRNA.fq | perl -e
1104 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1105 E2_R4_s_mRNA.fasta
1106 cat E2_R5_s_mRNA.fq | perl -e
1107 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1108 E2_R5_s_mRNA.fasta
1109 cat E2_R6_s_mRNA.fq | perl -e
1110 '$i=0;while(<>){if(/^@/)&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3;$i++;}}' >
1111 E2_R6_s_mRNA.fasta
1112
1113 rm *s_mRNA.fq
1114
1115 The appropriate database was downloaded from www.uniprot.org on March 6th, 2018 by either (1)
1116 searching for ec:* or (2) downloading the uniprot-all database. Both the TrEMBL and SwissProt
1117 sequences were included in the database. The format of the downloaded files was given in three
1118 columns: protein ID, EC#, and fasta sequence. After the download was complete, the files were
1119 unzipped. The database was processed as follows:

```

```

1120
1121 sed -i -e 's/^>/' NEW_EC_Library.tab
1122 sed -i -e 's/\t/\n/2' NEW_EC_Library.tab
1123 sed -i -e 's/ /_/g' NEW_EC_Library.tab
1124 sed -i -e 's/\t/\?/g' NEW_EC_Library.tab
1125 wc -l NEW_EC_Library.tab
1126 tail -n+3 NEW_EC_Library.tab > NEW_EC_Library.fasta
1127
1128 The DIAMOND software was used to run the annotation. The first step of this annotation was to
1129 compile the database:
1130
1131 diamond makedb --in NEW_EC_Library.fasta -d EC_NEW
1132
1133 The reactors were annotated with a minimum bitscore cutoff of 50 into a tabular output format:
1134
1135 diamond blastx --db EC_NEW --threads 19 --out
1136 exp1R1_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R1_s_mRNA.fasta --max-
1137 target-seqs 1 --min-score 50
1138 diamond blastx --db EC_NEW --threads 19 --out
1139 exp1R2_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R2_s_mRNA.fasta --max-
1140 target-seqs 1 --min-score 50
1141 diamond blastx --db EC_NEW --threads 19 --out
1142 exp1R3_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R3_s_mRNA.fasta --max-
1143 target-seqs 1 --min-score 50
1144 diamond blastx --db EC_NEW --threads 19 --out
1145 exp1R4_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R4_s_mRNA.fasta --max-
1146 target-seqs 1 --min-score 50
1147 diamond blastx --db EC_NEW --threads 19 --out
1148 exp1R5_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R5_s_mRNA.fasta --max-
1149 target-seqs 1 --min-score 50
1150 diamond blastx --db EC_NEW --threads 19 --out
1151 exp1R6_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R6_s_mRNA.fasta --max-
1152 target-seqs 1 --min-score 50
1153
1154 diamond blastx --db EC_NEW --threads 19 --out
1155 exp2R1_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R1_s_mRNA.fasta --max-
1156 target-seqs 1 --min-score 50
1157 diamond blastx --db EC_NEW --threads 19 --out
1158 exp2R2_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R2_s_mRNA.fasta --max-
1159 target-seqs 1 --min-score 50
1160 diamond blastx --db EC_NEW --threads 19 --out
1161 exp2R3_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R3_s_mRNA.fasta --max-
1162 target-seqs 1 --min-score 50
1163 diamond blastx --db EC_NEW --threads 19 --out
1164 exp2R4_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R4_s_mRNA.fasta --max-
1165 target-seqs 1 --min-score 50
1166 diamond blastx --db EC_NEW --threads 19 --out
1167 exp2R5_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R5_s_mRNA.fasta --max-
1168 target-seqs 1 --min-score 50
1169 diamond blastx --db EC_NEW --threads 19 --out
1170 exp2R6_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R6_s_mRNA.fasta --max-
1171 target-seqs 1 --min-score 50
1172

```

```

1173 The files were further processed for downstream analyses:
1174
1175 sed -i 's/\?/\?_/g' *diamond_fullEC_NEW.out
1176 sed -i 's/\t/,/g' *diamond_fullEC_NEW.out
1177 sed -i 's/;/,/g' *diamond_fullEC_NEW.out
1178
1179 For the collation of the taxonomic origin of the individual read annotations, the first step was to
1180 collect the counts per unique Uniprot identification. This step was accomplished by processing
1181 within the Linux Ubuntu shell:
1182
1183 sed -i 's/,/_/\$/g' *diamond_fullEC_NEW.out
1184
1185 cat exp1R1_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1186 -e 's/^[ \t]*//' > exp1R1_counts_per_uniprotMarch2018.out
1187 cat exp1R2_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1188 -e 's/^[ \t]*//' > exp1R2_counts_per_uniprotMarch2018.out
1189 cat exp1R3_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1190 -e 's/^[ \t]*//' > exp1R3_counts_per_uniprotMarch2018.out
1191 cat exp1R4_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1192 -e 's/^[ \t]*//' > exp1R4_counts_per_uniprotMarch2018.out
1193 cat exp1R5_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1194 -e 's/^[ \t]*//' > exp1R5_counts_per_uniprotMarch2018.out
1195 cat exp1R6_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1196 -e 's/^[ \t]*//' > exp1R6_counts_per_uniprotMarch2018.out
1197
1198 cat exp2R1_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1199 -e 's/^[ \t]*//' > exp2R1_counts_per_uniprotMarch2018.out
1200 cat exp2R2_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1201 -e 's/^[ \t]*//' > exp2R2_counts_per_uniprotMarch2018.out
1202 cat exp2R3_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1203 -e 's/^[ \t]*//' > exp2R3_counts_per_uniprotMarch2018.out
1204 cat exp2R4_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1205 -e 's/^[ \t]*//' > exp2R4_counts_per_uniprotMarch2018.out
1206 cat exp2R5_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1207 -e 's/^[ \t]*//' > exp2R5_counts_per_uniprotMarch2018.out
1208 cat exp2R6_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
1209 -e 's/^[ \t]*//' > exp2R6_counts_per_uniprotMarch2018.out
1210
1211 The files were aggregated using the CondensedAllExperimentCountspersUniprot.R in R 3.5.1:
1212
1213 library("rtk"); packageVersion("rtk") # At time of implementation, 0.2.5.3
1214 library("data.table");packageVersion("data.table") # At time of implementation,
1215 1.10.4
1216 library('dplyr'); packageVersion("dplyr") # At time of implementation, 0.7.2
1217 library('tidyr'); packageVersion("tidyr") # At time of implementation, 0.6.3
1218
1219 #Read in the EC Table, the first is the count value "name of reactor" and the second
1220 is the uniprot-ECnumber combined "EC"
1221 exp1R1 <-
1222 as.data.table(read.table("exp1R1_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1223 " ")); setnames(exp1R1,c("exp1R1","EC"))

```

```

1224 exp1R2 <-
1225 as.data.table(read.table("exp1R2_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1226 " ")); setnames(exp1R2,c("exp1R2","EC"))
1227 exp1R3 <-
1228 as.data.table(read.table("exp1R3_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1229 " ")); setnames(exp1R3,c("exp1R3","EC"))
1230 exp1R4 <-
1231 as.data.table(read.table("exp1R4_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1232 " ")); setnames(exp1R4,c("exp1R4","EC"))
1233 exp1R5 <-
1234 as.data.table(read.table("exp1R5_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1235 " ")); setnames(exp1R5,c("exp1R5","EC"))
1236 exp1R6 <-
1237 as.data.table(read.table("exp1R6_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1238 " ")); setnames(exp1R6,c("exp1R6","EC"))
1239
1240 exp2R1 <-
1241 as.data.table(read.table("exp2R1_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1242 " ")); setnames(exp2R1,c("exp2R1","EC"))
1243 exp2R2 <-
1244 as.data.table(read.table("exp2R2_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1245 " ")); setnames(exp2R2,c("exp2R2","EC"))
1246 exp2R3 <-
1247 as.data.table(read.table("exp2R3_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1248 " ")); setnames(exp2R3,c("exp2R3","EC"))
1249 exp2R4 <-
1250 as.data.table(read.table("exp2R4_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1251 " ")); setnames(exp2R4,c("exp2R4","EC"))
1252 exp2R5 <-
1253 as.data.table(read.table("exp2R5_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1254 " ")); setnames(exp2R5,c("exp2R5","EC"))
1255 exp2R6 <-
1256 as.data.table(read.table("exp2R6_counts_per_uniprotMarch2018.out",header=FALSE, sep =
1257 " ")); setnames(exp2R6,c("exp2R6","EC"))
1258
1259 #Set the uniprot-ECnumber combined "EC" as the key
1260 setkey(exp1R1,"EC")
1261 setkey(exp1R2,"EC")
1262 setkey(exp1R3,"EC")
1263 setkey(exp1R4,"EC")
1264 setkey(exp1R5,"EC")
1265 setkey(exp1R6,"EC")
1266
1267 setkey(exp2R1,"EC")
1268 setkey(exp2R2,"EC")
1269 setkey(exp2R3,"EC")
1270 setkey(exp2R4,"EC")
1271 setkey(exp2R5,"EC")
1272 setkey(exp2R6,"EC")
1273
1274 #Merge the experiments
1275 merged_exp1 <- merge(merge(merge(merge(exp1R1, exp1R2, all=TRUE), exp1R3,
1276 all=TRUE), exp1R4, all=TRUE), exp1R5, all = TRUE), exp1R6, all = TRUE)
1277 merged_exp1_df <- as.data.frame(merged_exp1[,2:7])
1278

```

```

1279 merged_exp2 <- merge(merge(merge(merge(merge(exp2R1, exp2R2, all=TRUE), exp2R3,
1280 all=TRUE), exp2R4, all=TRUE), exp2R5, all = TRUE), exp2R6, all = TRUE)
1281 merged_exp2_df <- as.data.frame(merged_exp2[,2:7])
1282
1283 merged_all <- merge(merged_exp1, merged_exp2, all=TRUE)
1284 merged_all_df <- as.data.frame(merged_all[,1:13])
1285 merged_all_df[is.na(merged_all_df)] <- 0
1286
1287 #Split the UniprotID and EC
1288 output <- merged_all_df %>%
1289   separate(EC, c("UniprotID", "EC"), "\\?_")
1290
1291 write.csv(output,"CondensedTableUniprot.txt", sep = "\t")
1292
1293 The appropriate database was downloaded from www.uniprot.org on March 6th, 2018 by searching
1294 for ec:*. Both the TreMBL and SwissProt sequences were included in the database. The format of the
1295 downloaded files was given in multiple columns: uniprot ID and the subsequent levels of taxonomic
1296 annotation (super kingdom, kingdom, phylum, class, order, family, genus). After the download was
1297 complete, the files were unzipped. The downloaded file was processed to ensure that the first
1298 separator was a tab and the remaining separators were semi-colons:
1299
1300 awk -F '$\t' '{print $1,"\t",$4,";", $5,";", $6,";", $7,";", $8,";", $9,";", $10}'
1301 NEW_EC_Library.tab | sed 's/ \t/\t/g' > NEW_EC_uniprot.tab
1302
1303 The tab file was then converted into an SQLITE database for faster and more efficient processing
1304 downstream using the following TaxonomySQLiteCreationGenus.py python script:
1305
1306 import sqlite3
1307 import csv
1308 import sys
1309
1310 sqlite_file = 'Uniprot_Taxonomy_to_Genus.db' # name of the sqlite database file
1311 table_name2 = 'Uniprot_Taxonomy_to_Genus' # name of the table to be created
1312 id_field = 'id' # name of the column
1313 taxon_field = 'taxon' # name of the column
1314 field_type = 'TEXT' # column data type
1315
1316 # Connecting to the database file
1317 conn = sqlite3.connect(sqlite_file)
1318 c = conn.cursor()
1319
1320 # Creating a second table with 1 column and set it as PRIMARY KEY
1321 # note that PRIMARY KEY column must consist of unique values!
1322 c.execute('CREATE TABLE {tn} ({nf} {ft} PRIMARY KEY, {cf} {ft})'\
1323   .format(tn=table_name2, nf=id_field, ft=field_type, cf=taxon_field))
1324
1325 filename = sys.argv[-1]

```

```

1325 with open(filename, 'r') as f:
1326     reader = csv.reader(f, delimiter='\t')
1327     for row in reader:
1328         c.execute('INSERT INTO Uniprot_Taxonomy_to_Genus VALUES (?,?)', row)
1329
1330 # Committing changes and closing the connection to the database file
1331 conn.commit()
1332 conn.close()
1333
1334 Each EC number is processed individually from the condensed table:
1335
1336 grep -Pe 1\\.1\\.1\\.1[\\$\\"] CondensedTableUniprot.txt > 1.1.1.1_to_taxon_trans.txt
1337 grep -Pe 1\\.1\\.1\\.1\\.10[\\$\\"] CondensedTableUniprot.txt > 1.1.1.10_to_taxon_trans.txt
1338 grep -Pe 1\\.1\\.1\\.1\\.100[\\$\\"] CondensedTableUniprot.txt >
1339 1.1.1.100_to_taxon_trans.txt
1340 grep -Pe 1\\.1\\.1\\.1\\.101[\\$\\"] CondensedTableUniprot.txt >
1341 1.1.1.101_to_taxon_trans.txt
1342 (etc.)
1343
1344 python Taxonomy_Lookup.py 1.1.1.1_to_taxon_trans.txt > 1.1.1.1_taxon.txt
1345 python Taxonomy_Lookup.py 1.1.1.10_to_taxon_trans.txt > 1.1.1.10_taxon.txt
1346 python Taxonomy_Lookup.py 1.1.1.100_to_taxon_trans.txt > 1.1.1.100_taxon.txt
1347 python Taxonomy_Lookup.py 1.1.1.101_to_taxon_trans.txt > 1.1.1.101_taxon.txt
1348 (etc.)
1349
1350 Where the Taxonomy_lookup.py script is as follows:
1351
1352 import sqlite3
1353 import csv
1354 import sys
1355
1356 sqlite_file = 'Uniprot_Taxonomy_to_Genus.db' # name of the sqlite database file
1357 table_name2 = 'Uniprot_Taxonomy_output' # name of the table to be created
1358 id_field = 'id' # name of the column
1359 taxon_field = 'taxon' # name of the column
1360 field_type = 'TEXT' # column data type
1361
1362 # Connecting to the database file
1363 conn = sqlite3.connect(sqlite_file)
1364 c = conn.cursor()
1365
1366 # Creating a second table with 1 column and set it as PRIMARY KEY
1367 # note that PRIMARY KEY column must consist of unique values!
1368
1369 filename = sys.argv[-1]
1370
1371 with open(filename, 'r') as f:
1372     reader = csv.reader(f, delimiter=',')
1373     for row in reader:
1374
1375         c.execute('SELECT * FROM Uniprot_Taxonomy_to_Genus WHERE id=?',(row[1],))
1376         xs = c.fetchall()
1377         for x in xs:
1378             print x[1] + '\t' + row[3] + '\t' + row[4] + '\t' + row[5] + '\t' + row[6] + '\t' + row[7] +
1379 '\t' + row[8] + '\t' + row[9] + '\t' + row[10] + '\t' + row[11] + '\t' + row[12] + '\t' + row[13] + '\t'
1380 + row[14]
1381
1382 # Committing changes and closing the connection to the database file
1383 conn.commit()
1384 conn.close()
1385
1386 The results were collated using the CondenseTaxonTable_fourths.R R-Script:

```

```

1387
1388 #=====
1389 # Load the required libraries and print the version numbers
1390 #=====
1391 library("plyr");           packageVersion("plyr")           # At time of implementation, 1.8.4
1392 library("ggplot2");       packageVersion("ggplot2")       # At time of implementation, 2.2.1
1393 library("RColorBrewer");  packageVersion("RColorBrewer")  # At time of implementation, 1.1.2
1394 library("rtk");           packageVersion("rtk")           # At time of implementation, 0.2.5.3
1395 library("data.table");    packageVersion("data.table")    # At time of implementation, 1.10.4
1396 library("vegan");         packageVersion("vegan")         # At time of implementation, 2.4.3
1397 library("dplyr");         packageVersion("dplyr")         # At time of implementation, 0.7.2
1398 library("tidyr");         packageVersion("tidyr")         # At time of implementation, 0.6.3
1399
1400 #=====
1401 # Read in the list of EC Names
1402 #=====
1403 third_names <- read.table("Fourth_Taxon_Table_Names.txt", sep="\t", quote = "", row.names = NULL,
1404 stringsAsFactors = FALSE, comment.char = "")
1405
1406 #=====
1407 # Setup Summary Matrix that will hold the results and be printed at the end
1408 #=====
1409 summary <- matrix(ncol = length(third_names[,1]), nrow = 48)
1410
1411 #=====
1412 # Loopover all EC Numbers
1413 #=====
1414 for (i in 1:length(third_names[,1])){
1415   print(paste("Analyzing ",third_names[i,1]))
1416   if (file.info(third_names[i,1])$size > 0){
1417     #=====
1418     # Read in the EC Group
1419     #=====
1420     tab <- read.table(third_names[i,1], sep="\t", quote = "", row.names = NULL, stringsAsFactors = FALSE,
1421 comment.char = "")
1422     grepl('^;',tab[,1])
1423     #=====
1424     # Condense into unique taxons (sum) and write the condensed file
1425     #=====
1426     tab2 <- ddply(tab, .(V1), numcolwise(sum))
1427     write.table(tab2, paste("condensed",third_names[i,1]), sep='$')
1428     #=====
1429     # Parse the tab2 dataframe into the exp1 and exp2 components for housekeeping
1430     #=====
1431     tab2exp1_df <- tab2[,2:7]
1432     tab2exp2_df <- tab2[,8:13]
1433     #=====
1434     # Establish the minimum threshold to be considered detected
1435     #=====
1436     threshold <-
1437 min(c(max(sum(tab2exp1_df[,1]),1),max(sum(tab2exp1_df[,2]),1),max(sum(tab2exp1_df[,3]),1),max(sum(tab2exp1
1438 _df[,4]),1),max(sum(tab2exp1_df[,5]),1),max(sum(tab2exp1_df[,6]),1),
1439
1440 max(sum(tab2exp2_df[,1]),1),max(sum(tab2exp2_df[,2]),1),max(sum(tab2exp2_df[,3]),1),max(sum(tab2exp2_df[,4
1441 ],1),max(sum(tab2exp2_df[,5]),1),max(sum(tab2exp2_df[,6]),1)
1442 ))
1443     #=====
1444     # Calculate the diversity metrics (richness, Shannon, total number) for each of the looped over third
1445 orders
1446     # process experiment 1 #=====
1447     summary[1,i] <- sum(tab2exp1_df[,1])/sum(tab2exp1_df[,1]) >= 1/threshold
1448     summary[2,i] <- sum(tab2exp1_df[,2])/sum(tab2exp1_df[,2]) >= 1/threshold
1449     summary[3,i] <- sum(tab2exp1_df[,3])/sum(tab2exp1_df[,3]) >= 1/threshold
1450     summary[4,i] <- sum(tab2exp1_df[,4])/sum(tab2exp1_df[,4]) >= 1/threshold
1451     summary[5,i] <- sum(tab2exp1_df[,5])/sum(tab2exp1_df[,5]) >= 1/threshold
1452     summary[6,i] <- sum(tab2exp1_df[,6])/sum(tab2exp1_df[,6]) >= 1/threshold
1453     summary[13,i] <- -sum(tab2exp1_df[,1]/sum(tab2exp1_df[,1])*log2(tab2exp1_df[,1]/sum(tab2exp1_df[,1])),
1454 na.rm = TRUE)
1455     summary[14,i] <- -sum(tab2exp1_df[,2]/sum(tab2exp1_df[,2])*log2(tab2exp1_df[,2]/sum(tab2exp1_df[,2])),
1456 na.rm = TRUE)

```



```

1457     summary[15,i] <- -sum(tab2exp1_df[,3]/sum(tab2exp1_df[,3])*log2(tab2exp1_df[,3]/sum(tab2exp1_df[,3])),
1458     na.rm = TRUE)
1459     summary[16,i] <- -sum(tab2exp1_df[,4]/sum(tab2exp1_df[,4])*log2(tab2exp1_df[,4]/sum(tab2exp1_df[,4])),
1460     na.rm = TRUE)
1461     summary[17,i] <- -sum(tab2exp1_df[,5]/sum(tab2exp1_df[,5])*log2(tab2exp1_df[,5]/sum(tab2exp1_df[,5])),
1462     na.rm = TRUE)
1463     summary[18,i] <- -sum(tab2exp1_df[,6]/sum(tab2exp1_df[,6])*log2(tab2exp1_df[,6]/sum(tab2exp1_df[,6])),
1464     na.rm = TRUE)
1465     summary[25,i] <- sum(tab2exp1_df[,1])
1466     summary[26,i] <- sum(tab2exp1_df[,2])
1467     summary[27,i] <- sum(tab2exp1_df[,3])
1468     summary[28,i] <- sum(tab2exp1_df[,4])
1469     summary[29,i] <- sum(tab2exp1_df[,5])
1470     summary[30,i] <- sum(tab2exp1_df[,6])
1471     #=====
1472     # Search for those reads originating from bacteria
1473     #=====
1474     summary[37,i] <- sum(tab2exp1_df[,1]*grepl('Bacteria ',tab2[,1]))
1475     summary[38,i] <- sum(tab2exp1_df[,2]*grepl('Bacteria ',tab2[,1]))
1476     summary[39,i] <- sum(tab2exp1_df[,3]*grepl('Bacteria ',tab2[,1]))
1477     summary[40,i] <- sum(tab2exp1_df[,4]*grepl('Bacteria ',tab2[,1]))
1478     summary[41,i] <- sum(tab2exp1_df[,5]*grepl('Bacteria ',tab2[,1]))
1479     summary[42,i] <- sum(tab2exp1_df[,6]*grepl('Bacteria ',tab2[,1]))
1480     #=====
1481     # process experiment 2 #=====
1482     summary[7,i] <- sum(tab2exp2_df[,1]/sum(tab2exp2_df[,1]) >= 1/threshold)
1483     summary[8,i] <- sum(tab2exp2_df[,2]/sum(tab2exp2_df[,2]) >= 1/threshold)
1484     summary[9,i] <- sum(tab2exp2_df[,3]/sum(tab2exp2_df[,3]) >= 1/threshold)
1485     summary[10,i] <- sum(tab2exp2_df[,4]/sum(tab2exp2_df[,4]) >= 1/threshold)
1486     summary[11,i] <- sum(tab2exp2_df[,5]/sum(tab2exp2_df[,5]) >= 1/threshold)
1487     summary[12,i] <- sum(tab2exp2_df[,6]/sum(tab2exp2_df[,6]) >= 1/threshold)
1488     summary[19,i] <- -sum(tab2exp2_df[,1]/sum(tab2exp2_df[,1])*log2(tab2exp2_df[,1]/sum(tab2exp2_df[,1])),
1489     na.rm = TRUE)
1490     summary[20,i] <- -sum(tab2exp2_df[,2]/sum(tab2exp2_df[,2])*log2(tab2exp2_df[,2]/sum(tab2exp2_df[,2])),
1491     na.rm = TRUE)
1492     summary[21,i] <- -sum(tab2exp2_df[,3]/sum(tab2exp2_df[,3])*log2(tab2exp2_df[,3]/sum(tab2exp2_df[,3])),
1493     na.rm = TRUE)
1494     summary[22,i] <- -sum(tab2exp2_df[,4]/sum(tab2exp2_df[,4])*log2(tab2exp2_df[,4]/sum(tab2exp2_df[,4])),
1495     na.rm = TRUE)
1496     summary[23,i] <- -sum(tab2exp2_df[,5]/sum(tab2exp2_df[,5])*log2(tab2exp2_df[,5]/sum(tab2exp2_df[,5])),
1497     na.rm = TRUE)
1498     summary[24,i] <- -sum(tab2exp2_df[,6]/sum(tab2exp2_df[,6])*log2(tab2exp2_df[,6]/sum(tab2exp2_df[,6])),
1499     na.rm = TRUE)
1500     summary[31,i] <- sum(tab2exp2_df[,1])
1501     summary[32,i] <- sum(tab2exp2_df[,2])
1502     summary[33,i] <- sum(tab2exp2_df[,3])
1503     summary[34,i] <- sum(tab2exp2_df[,4])
1504     summary[35,i] <- sum(tab2exp2_df[,5])
1505     summary[36,i] <- sum(tab2exp2_df[,6])
1506     summary[43,i] <- sum(tab2exp2_df[,1]*grepl('Bacteria ',tab2[,1]))
1507     summary[44,i] <- sum(tab2exp2_df[,2]*grepl('Bacteria ',tab2[,1]))
1508     summary[45,i] <- sum(tab2exp2_df[,3]*grepl('Bacteria ',tab2[,1]))
1509     summary[46,i] <- sum(tab2exp2_df[,4]*grepl('Bacteria ',tab2[,1]))
1510     summary[47,i] <- sum(tab2exp2_df[,5]*grepl('Bacteria ',tab2[,1]))
1511     summary[48,i] <- sum(tab2exp2_df[,6]*grepl('Bacteria ',tab2[,1]))
1512     } else {summary[1:48,i] <- 0}
1513     }
1514     #=====
1515     # Write the full final table
1516     #=====
1517     write.table(t(summary), "Summary of all Fourths with Count Rerun March 12 2018 at least one read with
1518     bacteria fraction.txt")
1519

```

```

1520 Supplemental File 3 : Sequencing Batch Reactor Monod model for multiple-species
1521
1522 # The following is an R script that can be run on R v 3.0 or higher. Only the package deSolve
1523 is required.
1524
1525 library(deSolve)
1526 #=====
1527 # Set the ranges of the ecological umax and Ks values
1528 #=====
1529 u_eco_max <- 9.8
1530 u_eco_min <- 0.2
1531
1532 #=====
1533 # Set the bulk Ks value
1534 #=====
1535 Ks_bulk <- 50
1536
1537 #=====
1538 # Select MRTs to use in the differential equations, establish certain required variables.
1539 # in this case, MRT of 1 day is used as the example.
1540 #=====
1541 RW1 <- c(); j <- 1; diversity <- c()
1542
1543 u_bulk = 5
1544
1545 # Use only one set of the u_s below for the solution carried forward, comment out all others
1546 with #
1547
1548 # Growth Rates for MRT 1 day
1549 u9 = 4.626275;u8 = 4.719709;u7 = 4.813127;u6 = 4.906577;u5 = u_bulk;u4 = 5.093445;u3 =
1550 5.186879;u2 = 5.280314;u1 = 5.373748
1551
1552 # Growth Rates for MRT 3 day
1553 # u9 = 4.054179;u8 = 4.290640;u7 = 4.527102;u6 = 4.763563;u5 = u_bulk;u4 = 5.236486;u3 =
1554 5.472947;u2 = 5.709408;u1 = 5.945870
1555 # Growth Rates for MRT 5 day
1556 # u9 = 3.597233;u8 = 3.947921;u7 = 4.298609;u6 = 4.649298;u5 = u_bulk;u4 = 5.350675;u3 =
1557 5.701365;u2 = 6.052054;u1 = 6.402743
1558 # Growth Rates for MRT 7 day
1559 # u9 = 3.288019;u8 = 3.716036;u7 = 4.144053;u6 = 4.572071;u5 = u_bulk;u4 = 5.428105;u3 =
1560 5.856122;u2 = 6.284140;u1 = 6.712157
1561 # Growth Rates for MRT 10 day
1562 # u9 = 2.912718;u8 = 3.434623;u7 = 3.956528;u6 = 4.478434;u5 = u_bulk;u4 = 5.522244;u3 =
1563 6.044149;u2 = 6.566055;u1 = 7.087960
1564 # Growth Rates for MRT 15 day
1565 # u9 = 2.492964;u8 = 3.119928;u7 = 3.746893;u6 = 4.373858;u5 = u_bulk;u4 = 5.627788;u3 =
1566 6.254753;u2 = 6.881717;u1 = 7.508682
1567
1568 Ks9 = Ks_bulk;Ks8 = Ks_bulk;Ks7 = Ks_bulk;Ks6 = Ks_bulk;Ks5 = Ks_bulk;Ks4 = Ks_bulk;Ks3 =
1569 Ks_bulk;Ks2 = Ks_bulk;Ks1 = Ks_bulk
1570 b1=0.2; b2=0.1775 ;b3=0.155; b4=0.1325; b5=0.11; b6=0.0875; b7=0.065; b8=0.0425; b9=0.02
1571
1572 #=====
1573 # Set the parameters of the SBR
1574 #=====
1575 Volume = 12 # Volume of reactor
1576 days_to_model = 36 # Time to run the integration over
1577 sbr_seq = 1/6 # Given in days
1578 V_withdrawn = 4 # in Liters
1579 Slin = 250 # Substrate inflow rate
1580 Ys1S1 = 0.3 # Yield of bacterial growth (fixed)
1581 inflow_time = 20/(60*24) # Sequencing Batch Reactor Time step for the inflow

```

```

1582 outflow_time = 10/(60*24) # Sequencing Batch Reactor Time step for the outflow (drain)
1583 inflow_outflow_time = inflow_time + outflow_time
1584 settle_time = 30/(60*24) # Sequencing Batch Reactor Time step for the settling
1585 sludge_discharge_time = 10/(60*24) # Sequencing Batch Reactor Time step for the sludge removal
1586 integration_time_step = 0.5/(60*24) # Integration time step (set at useful interval)
1587 Inflow = 0
1588 Outflow = 0
1589
1590 #-----
1591 # In acknowledgement that the assumption that changing one unit in range normalized u_max
1592 # versus b_e values is equivalent is unlikely, a scaling parameter is established. When the
1593 # value is
1594 # <1, then the u_max contributes more to the combination of growth parameters. When the value
1595 # is >1,
1596 # the b_e contributes more. In this case, the value is set to 0.25 to assume
1597 # that the u_max value range contributes more than the b_e. Values of 0.1
1598 # to 0.8 were explored, and the major influence was on the magnitude of the difference in
1599 # richness and
1600 # diversity across the MRT gradient, not the shape of the underlying relationship. Therefore,
1601 # 0.25 was
1602 # selected for efficient computational time. As the value approaches 1 or exceeds one, changes
1603 # in richness
1604 # and diversity is minimized because the b_e range is saturated.
1605 #-----
1606 Scaling_Factor = 0.25
1607
1608 #-----
1609 # How long the reactors are operated for (community equilibrium time)
1610 times <- seq(0, days_to_model, by = integration_time_step)
1611
1612 #=====
1613 # Parse the differential equations into a function entitles "SBR_Diff_Iter". Iterate to find a
1614 # parametrically acceptable solution for the growth parameters that lead to persistence.
1615 #=====
1616 repeat{ SBR_Diff_Iter <- function(t, state, parameters){
1617   with(as.list(c(state, parameters)),{
1618     Qfill <- inflow_on_func(t)
1619     Qclearwater_drain <- outflow_on_func(t)
1620     Qmixed_drain <- waste_rate_on_func(t)
1621     dV <- Qfill - Qclearwater_drain - Qmixed_drain
1622     dX1 <- (-X1/V * Qmixed_drain + u1 * (So)/(Ks1 + So) * X1 - b1 * X1)
1623     dX2 <- (-X2/V * Qmixed_drain + u2 * (So)/(Ks2 + So) * X2 - b2 * X2)
1624     dX3 <- (-X3/V * Qmixed_drain + u3 * (So)/(Ks3 + So) * X3 - b3 * X3)
1625     dX4 <- (-X4/V * Qmixed_drain + u4 * (So)/(Ks4 + So) * X4 - b4 * X4)
1626     dX5 <- (-X5/V * Qmixed_drain + u5 * (So)/(Ks5 + So) * X5 - b5 * X5)
1627     dX6 <- (-X6/V * Qmixed_drain + u6 * (So)/(Ks6 + So) * X6 - b6 * X6)
1628     dX7 <- (-X7/V * Qmixed_drain + u7 * (So)/(Ks7 + So) * X7 - b7 * X7)
1629     dX8 <- (-X8/V * Qmixed_drain + u8 * (So)/(Ks8 + So) * X8 - b8 * X8)
1630     dX9 <- (-X9/V * Qmixed_drain + u9 * (So)/(Ks9 + So) * X9 - b9 * X9)
1631
1632     dmass <- dX1 + dX2 + dX3 + dX4 + dX5 + dX6 + dX7 + dX8 + dX9
1633
1634     dSo <- (Qfill*SS1in/V - Qclearwater_drain*So/V - So/V * Qmixed_drain - u1 * (So)/(Ks1 +
1635 (So)) * (X1) / Yield -
1636         u2 * (So)/(Ks2 + (So)) * (X2) / Yield -
1637         u3 * (So)/(Ks3 + (So)) * (X3) / Yield -
1638         u4 * (So)/(Ks4 + (So)) * (X4) / Yield -
1639         u5 * (So)/(Ks5 + (So)) * (X5) / Yield -
1640         u6 * (So)/(Ks6 + (So)) * (X6) / Yield -
1641         u7 * (So)/(Ks7 + (So)) * (X7) / Yield -
1642         u8 * (So)/(Ks8 + (So)) * (X8) / Yield -
1643         u9 * (So)/(Ks9 + (So)) * (X9) / Yield)

```

```

1644
1645     list(c(dV, dSo, dX1,dX2,dX3,dX4,dX5,dX6,dX7,dX8,dX9, dmass))
1646   })
1647 }
1648
1649 parameters500 <- c( SS1in = S1in, SBR = sbr_seq,
1650                    u1 = u1, u2 = u2, u3 = u3, u4 = u4, u5 = u5,
1651                    Ks1 = Ks1, Ks2 = Ks2, Ks3 = Ks3, Ks4 = Ks4, Ks5 = Ks5,
1652                    u6 = u6, u7 = u7, u8 = u8, u9 = u9,
1653                    Ks6 = Ks6, Ks7 = Ks7, Ks8 = Ks8, Ks9 = Ks9,
1654                    Yield = Ys1S1,
1655                    b1=b1,b2=b2, b3=b3, b4=b4, b5=b5, b6=b6, b7=b7, b8=b8, b9=b9)
1656
1657 #####
1658 # Set the initial State
1659 #####
1660 n = 9
1661 num_spec <- as.numeric(seq(1,1, length.out = n))
1662 state <- c(V= Volume, So = 250, X = num_spec*100/8, mass = 100)
1663
1664 #####
1665 # Establish the inflow/outflow/waste triggers to describe the SBR system, turn on the
1666 # appropriate functional set that
1667 # matches the selected MRT above.
1668 #####
1669 #-----
1670 #For 1d MRT
1671 #-----
1672 inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1673 inflow_time)/integration_time_step + 1),
1674                                rep(4/inflow_time,
1675 (inflow_time)/integration_time_step + 1)), days_to_model/sbr_seq),0)))
1676
1677 inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1678
1679 outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1680 inflow_outflow_time)/integration_time_step + 1), rep(2/outflow_time,
1681 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
1682 days_to_model/sbr_seq),0)))
1683
1684 outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1685
1686 waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1687 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1688 rep(2/sludge_discharge_time, sludge_discharge_time/integration_time_step + 1),
1689                                rep(0, (inflow_outflow_time
1690 + settle_time)/integration_time_step + 1)), days_to_model/sbr_seq),0)))
1691
1692 waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1693
1694 #-----
1695 #For 3d MRT
1696 #-----
1697
1698 #inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1699 inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
1700 + 1)), days_to_model/sbr_seq),0)))
1701 #inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1702
1703 #outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1704 inflow_outflow_time)/integration_time_step + 1), rep(2/outflow_time,

```

```

1705 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1706 1), rep(0, 480), rep(0, 480)), days_to_model*2, 0)))
1707 #outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1708
1709 #waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1710 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1711 rep(4/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1712 (inflow_outflow_time + settle_time)/integration_time_step + 1)), days_to_model/sbr_seq), 0)))
1713 #waste_rate_outflow <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1714 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1715 rep(2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1716 (inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0, 480), rep(0, 480)),
1717 days_to_model*2, 0)))
1718 #waste_rate_on <- waste_rate_on - waste_rate_outflow
1719 #waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1720
1721 #-----
1722 #For 5d MRT
1723 #-----
1724 #inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1725 inflow_time)/integration_time_step + 1), rep(4/inflow_time, (inflow_time)/integration_time_step
1726 + 1)), days_to_model/sbr_seq), 0)))
1727 #inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1728
1729 #outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1730 inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
1731 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
1732 days_to_model/sbr_seq), 0)))
1733 #outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
1734 - inflow_outflow_time)/integration_time_step + 1), rep(1.2/outflow_time,
1735 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1736 1), rep(0, 480), rep(0, 480)), days_to_model*2, 0)))
1737 #outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
1738 #outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1739
1740 #waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1741 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1742 rep(1.2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1743 (inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0, 480), rep(0, 480)),
1744 days_to_model*2, 0)))
1745 #waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1746
1747 #-----
1748 #For 7d MRT
1749 #-----
1750 #inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1751 inflow_time)/integration_time_step + 1), rep(4/inflow_time, (inflow_time)/integration_time_step
1752 + 1)), days_to_model/sbr_seq), 0)))
1753 #inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1754
1755 #outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1756 inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
1757 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
1758 days_to_model/sbr_seq), 0)))
1759 #outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
1760 - inflow_outflow_time)/integration_time_step + 1), rep(1.7/outflow_time,
1761 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1762 1), rep(0, 480*5)), days_to_model), 0)))
1763 #outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
1764 #outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1765

```

```

1766 #waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1767 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1768 rep(1.7/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1769 (inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
1770 days_to_model),0)))
1771 #waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1772
1773 #-----
1774 #For 10d MRT
1775 #-----
1776 #inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1777 inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
1778 + 1)), days_to_model/sbr_seq),0)))
1779 #inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1780
1781 #outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1782 inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
1783 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
1784 days_to_model/sbr_seq),0)))
1785 #outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
1786 - inflow_outflow_time)/integration_time_step + 1), rep(1.2/outflow_time,
1787 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1788 1),rep(0,480*5)), days_to_model),0)))
1789 #outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
1790 #outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1791
1792 #waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1793 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1794 rep(1.2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1795 (inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
1796 days_to_model),0)))
1797 #waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1798
1799 #-----
1800 #For 15d MRT
1801 #-----
1802 #inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1803 inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
1804 + 1)), days_to_model/sbr_seq),0)))
1805 #inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")
1806
1807 #outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1808 inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
1809 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
1810 days_to_model/sbr_seq),0)))
1811 #outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
1812 - inflow_outflow_time)/integration_time_step + 1), rep(0.8/outflow_time,
1813 (outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1814 1),rep(0,480*5)), days_to_model),0)))
1815 #outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
1816 #outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
1817
1818 #waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
1819 inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
1820 rep(0.8/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
1821 (inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
1822 days_to_model),0)))
1823 #waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")
1824
1825 #=====
1826 # Run the differential equation
1827 #=====

```

```

1828 RW1 <- ode(y = state, times = times, func = SBR_Diff_Iter, parms = parameters500, method =
1829 "bdf")
1830
1831 output_error <- RW1[480*6*10,]/RW1[480*6*15,]
1832 abs_output_error <- abs(output_error-output_error[13])
1833 u9 = unname(u9+(output_error[12]-output_error[13])-0.5*(output_error[8]-output_error[13]),
1834 force =FALSE)
1835 u8 = unname(u8+(output_error[11]-output_error[13])-0.5*(output_error[8]-output_error[13]),
1836 force =FALSE)
1837 u7 = unname(u7+(output_error[10]-output_error[13])-0.5*(output_error[8]-output_error[13]),
1838 force =FALSE)
1839 u6 = unname(u6+(output_error[9]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
1840 =FALSE)
1841 u5 = u_bulk
1842 u4 = unname(u4+(output_error[7]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
1843 =FALSE)
1844 u3 = unname(u3+(output_error[6]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
1845 =FALSE)
1846 u2 = unname(u2+(output_error[5]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
1847 =FALSE)
1848 u1 = unname(u1+(output_error[4]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
1849 =FALSE)
1850
1851 u_s <- c(u1,u2,u3,u4,u5,u6,u7,u8,u9)
1852
1853 print(max(abs(output_error[4:13]-1)))
1854 print(u_s)
1855 plot(RW1, mfrow = c(3,5))
1856
1857 #Check whether the solution is in an acceptable error range
1858 if (! max(abs(output_error[4:13]-1)) > 0.0001){break}
1859 }
1860
1861 #=====
1862 # Plot the results of the optimization
1863 #=====
1864 plot(RW1, mfrow = c(3,5),xlim=c(14,15))
1865
1866 #=====
1867 # Establish the differential equations for a SBR AS system as a string to be parsed
1868 #=====
1869 b_s <- c(0.2, 0.1775, 0.155, 0.1325, 0.11, 0.0875, 0.065, 0.0425, 0.02)
1870 umax_s <- c(u1,u2,u3,u4,u5,u6,u7,u8,u9)
1871
1872 n = round(sqrt(((max(umax_s)-min(umax_s))/(u_eco_max-u_eco_min))^2+(0.18/0.18*
1873 Scaling_Factor)^2)*50)
1874 fit <- lm(umax_s~b_s)
1875
1876 b_range <- seq(min(b_s),max(b_s), length.out = n)
1877 umax_range <- fit$coefficients[2] * b_range + fit$coefficients[1] #+ fit2$coefficients[4] *
1878 rv$Kss1S1^3
1879
1880 dX_carb_eq <- paste("dX", 1:n, " <- (-X", 1:n, "/V * Qmixed_drain + u", 1:n,
1881 " * (So)/(Ks + So) * X", 1:n, " - b", 1:n, " * X", 1:n, ")", sep="")
1882
1883 dmass_eq <- paste("dmass <- ", paste("dX", 1:n, sep=" ",collapse=" + "), sep="")
1884
1885 dS_eq <- paste("dSo <- (Qfill*SS1in/V - Qclearwater_drain*So/V - So/V * Qmixed_drain - ",
1886 paste("u", 1:n, " * (So)/(Ks + So) * (X", 1:n, " ) / Yield", sep=" ",collapse="
1887 - "), ")", sep="")
1888

```

```

1889 state_list <- paste("list(c(dV, dSo, ", paste("dX",1:n, sep="", collapse = ","), ", dmass))",
1890 sep="", collapse = "")
1891
1892 write(c(dX_carb_eq,dmass_eq,dS_eq,state_list),"equation_output.txt")
1893
1894 #=====
1895 # Parse the differential equations into a function entitles "SBR_Diff"
1896 #=====
1897 # SBR_Diff <- function(t, state, parameters){
1898 #   with(as.list(c(state, parameters)),{
1899 #     Qfill <- inflow_on_func(t)
1900 #     Qclearweater_drain <- outflow_on_func(t)
1901 #     Qmixed_drain <- waste_rate_on_func(t)
1902 #     dV <- Qfill - Qmixed_drain - Qclearweater_drain
1903 #     eval(parse(text=dX_carb_eq))
1904 #     eval(parse(text=dmass_eq))
1905 #     eval(parse(text=dS_eq))
1906 #     eval(parse(text=state_list))
1907 #   })
1908 #}
1909 #=====
1910 # For large systems of equations, copy the text from the generated "equation_output" file and
1911 # attach below.
1912 # For larger n species, the system will take a substantial amount of time to run
1913 #=====
1914 SBR_Diff <- function(t, state, parameters){
1915   with(as.list(c(state, parameters)),{
1916     Qfill <- inflow_on_func(t)
1917     Qclearweater_drain <- outflow_on_func(t)
1918     Qmixed_drain <- waste_rate_on_func(t)
1919     dV <- Qfill - Qclearweater_drain - Qmixed_drain
1920     dX1 <- (-X1/V * Qmixed_drain + u1 * (So)/(Ks + So) * X1 - b1 * X1)
1921     dX2 <- (-X2/V * Qmixed_drain + u2 * (So)/(Ks + So) * X2 - b2 * X2)
1922     dX3 <- (-X3/V * Qmixed_drain + u3 * (So)/(Ks + So) * X3 - b3 * X3)
1923     dX4 <- (-X4/V * Qmixed_drain + u4 * (So)/(Ks + So) * X4 - b4 * X4)
1924     dX5 <- (-X5/V * Qmixed_drain + u5 * (So)/(Ks + So) * X5 - b5 * X5)
1925     dX6 <- (-X6/V * Qmixed_drain + u6 * (So)/(Ks + So) * X6 - b6 * X6)
1926     dX7 <- (-X7/V * Qmixed_drain + u7 * (So)/(Ks + So) * X7 - b7 * X7)
1927     dX8 <- (-X8/V * Qmixed_drain + u8 * (So)/(Ks + So) * X8 - b8 * X8)
1928     dX9 <- (-X9/V * Qmixed_drain + u9 * (So)/(Ks + So) * X9 - b9 * X9)
1929     dX10 <- (-X10/V * Qmixed_drain + u10 * (So)/(Ks + So) * X10 - b10 * X10)
1930     dX11 <- (-X11/V * Qmixed_drain + u11 * (So)/(Ks + So) * X11 - b11 * X11)
1931     dX12 <- (-X12/V * Qmixed_drain + u12 * (So)/(Ks + So) * X12 - b12 * X12)
1932     dX13 <- (-X13/V * Qmixed_drain + u13 * (So)/(Ks + So) * X13 - b13 * X13)
1933
1934     dmass <- dX1 + dX2 + dX3 + dX4 + dX5 + dX6 + dX7 + dX8 + dX9 + dX10 + dX11 + dX12 + dX13
1935
1936     dSo <- (Qfill*SS1in/V - Qclearweater_drain*So/V - So/V * Qmixed_drain - u1 * (So)/(Ks + So)
1937 *
1938           (X1) / Yield - u2 * (So)/(Ks + So) * (X2) / Yield - u3 * (So)/(Ks + So) * (X3) /
1939 Yield - u4 * (So)/(Ks + So) *
1940           (X4) / Yield - u5 * (So)/(Ks + So) * (X5) / Yield - u6 * (So)/(Ks + So) * (X6) /
1941 Yield - u7 * (So)/(Ks + So) *
1942           (X7) / Yield - u8 * (So)/(Ks + So) * (X8) / Yield - u9 * (So)/(Ks + So) * (X9) /
1943 Yield - u10 * (So)/(Ks + So) *
1944           (X10) / Yield - u11 * (So)/(Ks + So) * (X11) / Yield - u12 * (So)/(Ks + So) *
1945 (X12) / Yield - u13 * (So)/(Ks + So) *
1946           (X13) / Yield)
1947     list(c(dV, dSo, dX1,dX2,dX3,dX4,dX5,dX6,dX7,dX8,dX9,dX10,dX11,dX12,dX13, dmass)) })
1948 }
1949

```



```

1950 parameters_final <- c(b = b_range, SS1in = S1in, tcycle = sbr_seq, V_withdrawn_per_cycle =
1951 V_withdrawn,
1952                        u = umax_range, Ks = Ks_bulk, Yield = Ys1S1)
1953
1954 #####
1955 # Set the initial State
1956 #####
1957 num_spec <- as.numeric(seq(1,1, length.out = n))
1958 state <- c(V= Volume, So = 250, X = num_spec*100/n, mass = 100)
1959
1960 #####
1961 # Run the differential equation
1962 #####
1963 RW_final <- ode(y = state, times = times, func = SBR_Diff, parms = parameters_final)
1964
1965 #####
1966 # Calculate the Shannon diversity metric
1967 #####
1968 mass_i <- n + 4
1969 final_X <- n + 3
1970 diversity_time <- (days_to_model/integration_time_step-1)
1971 diversity <- sum(-1*RW_final[diversity_time,4:final_X]/(RW_final[diversity_time,mass_i]) *
1972                log(RW_final[diversity_time,4:final_X]/(RW_final[diversity_time,mass_i])))
1973 TSS <- RW_final[diversity_time,mass_i]
1974
1975 #####
1976 # Plot the results, NOTE - not volume normalized here
1977 #####
1978 plot(RW_final)
1979 plot(RW_final, mfrow = c(3,5),xlim=c(29,30))
1980 plot(RW_final, mfrow = c(3,5),xlim=c(29,30),ylim=c(10,13))
1981
1982 #####
1983 # Solve the same system of equations for an initial spiked state of So = 500 mg/L to determine
1984 # the theoretical instantaneous oxygen uptake rate (OUR).
1985 #####
1986 Spike_state <- c(V= Volume, So = 500, X = as.numeric(RW_final[diversity_time,4:final_X]), mass
1987 = TSS)
1988 RW_spike <- ode(y = Spike_state, times = times[1:480], func = SBR_Diff, parms =
1989 parameters_final)
1990 init_time <- RW_spike[1:20,1]
1991 init_subs <- RW_spike[1:20,3]
1992 fit_spike <- lm(init_subs~init_time)
1993 fit_spike
1994 fit_spike$coefficients[2]/TSS
1995
1996 sum(as.numeric(RW_final[diversity_time,4:final_X])*umax_range)/TSS
1997

```

1998 **Supplemental Materials 1** : Details of RNA and DNA isolation, handling, and sequencing protocols
1999
2000 **Supplemental Materials 2** : Equivalent model solution for a continuously-stirred tank reactor (CSTR)
2001 with R code
2002

2003

2004

Supplemental Materials 1 : Details of RNA and DNA isolation, handling, and sequencing protocols

RNA/DNA Isolation

In total, 10-mL acidic phenol-saturated water (pH 4.3; Sigma-Aldrich) was added to the frozen cell pellets. Thereof, 1 mL was transferred to a 2-mL tube containing 0.1-mm zirconia/silica beads and combined with 250- μ L pH 5.1 buffer (0.4-mL 0.5 M EDTA, molecular biology grade (Invitrogen), 0.33-mL 3-M sodium acetate (pH 5.1), molecular biology grade (Sigma-Aldrich), 19.3-mL DEPC-treated water) and 100- μ L 10% SDS (Sigma-Aldrich). After incubation at 65°C for 2 min, the samples were bead beat at max speed for 2 min (MM300 TissueLyser, Qiagen Retsch), incubated at 65°C for 8 min, bead beat at max speed for 2 min, and centrifuged at 14,000 \times g at 4°C for 5 min. The top layer was preserved for further processing.

Initially, 125 μ L of phenol-saturated water (pH 4.3; Sigma-Aldrich) and 125 μ L of chloroform:isoamylalcohol (24:1; Sigma-Aldrich) was added to the sample, and the mixture was shaken for 3 min, let stand for 3 min, and then centrifuged at 14,000 \times g at 4°C for 3 min. The top aqueous layer was transferred, and 100 μ L of phenol and 100 μ L of chloroform:isoamylalcohol (24:1) was added. The mixture was shaken for 3 min, let stood for 3 min, and then centrifuged at 14,000 \times g at 4°C for 3 min. The top aqueous layer was transferred, and 200 μ L of chloroform:isoamylalcohol was added. The mixture was shaken for 3 min, let stood for 3 min, and then centrifuged at 14,000 \times g at 4°C for 3 min. The top layer was transferred, and 0.5 volume of 7.5-M ammonium acetate and 2 volumes of 100% ethanol was added to precipitate the pellet. The sample was mixed, spun down, incubated at -20°C for at least 2 hours, and centrifuged at 21,000 \times g for 30 min at 4°C. After decanting the supernatant, the sample was rinsed with 200 μ L of 80% ethanol and centrifuged at 21,000 \times g for 10 min at 4°C. After decanting the supernatant, the sample was vacuum centrifuged for 2-3 min. The RNA pellet was re-suspended with DEPC-treated RNase-free water to a total of 25 μ L. Total nucleic acid concentration was measured on a Nanodrop (Invitrogen) or Qubit (Invitrogen), and aliquots of the samples were frozen at -80°C.

The protocol for extracting DNA followed the same procedure outlined above with the substitutions of pH 7.0 instead of pH 4.3 phenol-saturated water (Sigma Aldrich). The samples were quantified using Nanodrop (Invitrogen) or Qubit (Invitrogen), and the final samples were further cleaned using the DNA PowerCleanup PRO Kit (Qiagen) following the manufacturer's instructions. The purified DNA was frozen at -20°C until further use.

The RNA samples were further cleaned using the TURBO DNase Kit. and the MoBio RNA Pro Clean-Up Kit (MoBio) following the manufacturer's protocols. After reverse-transcription using the Superscript III kit (Invitrogen) the quality of the RNA was determined using a Bioanalyzer 2000 (Agilent Technologies) or TapeStation (Agilent Technologies), and the samples were stored at -80°C until further use.

16S Library Preparation and Sequencing

The initial polymerase chain reaction (PCR) consisted of (in triplication for each sample) 12.5 µl of 2X KAPA HiFi Hot Start Ready Mix, 0.75 µl forward primer (10 µM), 0.75 µl reverse primer (10 µM), 1.25 µl of DMSO, 9 µl of PCR grade water, and 1 µl of template DNA. Two sets of 16S rRNA primers (Integrated DNA Technologies, Inc., Skokie, Illinois, USA,) were used to amplify either the sample cDNA or gDNA to minimize the potential for probe bias [Guo et al., 2013]. The details of primers B1 and B2 are detailed in Supplemental Table 1. The PCR was run on a thermal cycler using the following program: 95°C for 5 min; cycles of 98°C for 20 seconds, 53°C for 15 seconds, and 72°C for 15 seconds (18 and 20 cycles for B1 and B2, respectively); 72°C for 5 minutes. The samples were then cleaned using AMPure Beads following the manufacturer's instructions. Nextera XT index primers (N7XX and S5XX; Illumina) were attached to the amplicons in a subsequent PCR: 25 µl of 2X KAPA HiFi, 5 µl each of Nextera XT index primer 1 and 2, and 15 µl of the cleaned amplicon run at 95°C for 3 min; 9 cycles of 95°C for 30 seconds, 55°C for 30 seconds, and 72°C for 30 seconds; 72°C for 5 minutes. The samples were again cleaned using AMPure Beads following the manufacturer's instructions. The samples were

2057 quantified and qualified using a Nanodrop (Invitrogen), Qubit (ThermoFisher), and Tapestation (Agilent).
2058 Samples were then normalized and pooled. Subsequently, the samples were sequenced using the PE 300
2059 method on a MiSeq platform (Illumina) at the Genomics Diversity Center at the ETH in Zurich,
2060 Switzerland. The raw data is available at EMBL-EBI under the study number ERP024418.

2061

2062

2063

2064

2065

2066

2067

2068

2069

2070

2071

2072

2073

2074

2075

2076

2077

2078

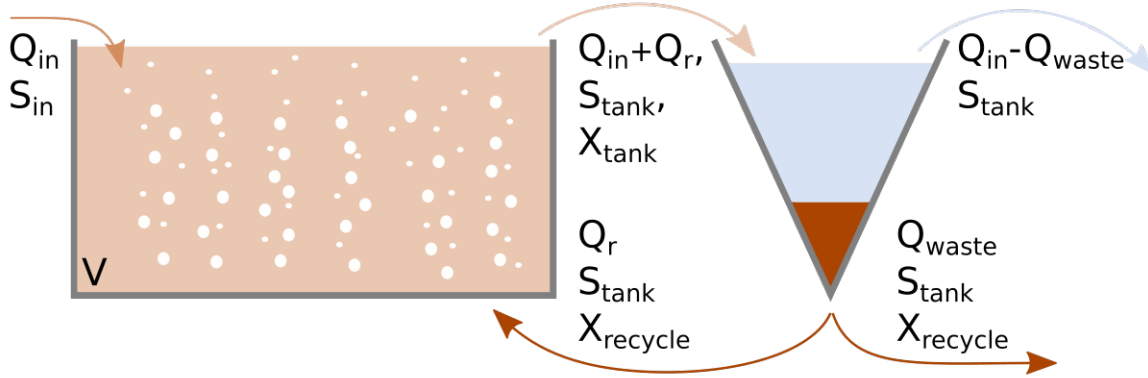
2079

2080

2081

Supplemental Materials 2 : Equivalent model solution for a continuously-stirred tank reactor (CSTR)

with R code



SM2 Figure 1. Diagram of the continuously stirred tank reactor (CSTR). Q_{waste} and $Q_{recycle}$ are the waste and recycle flow rates, respectively, and all other parameters are defined in the main text. Q_{waste} is determined by the MRT.

To establish the continuously stirred tank reactor model the S_{min}^* Equation 1 in the main text can be rearranged to solve for either the u_{max} or b_e parameter:

$$\mu_{max} = \frac{\left(1 + b_e * MRT * \left(1 + \frac{S^*}{K_S}\right)\right)^{\frac{K_S}{S^*} + 1}}{MRT} \quad \text{SM Eq. 1}$$

$$b_e = \frac{\frac{S^*}{K_S}(\mu_{max} * MRT - 1) - 1}{MRT \left(1 + \frac{S^*}{K_S}\right)} \quad \text{SM Eq. 2}$$

where all parameters have been previously described in the main text. As the MRT increases, these parameters are checked that they satisfy the ecological maximum and minimum defined; when they violate a parameter, they are appropriately replaced.

Conventional differential equations are established to describe the CSTR with recycle:

$$\frac{dS}{dt} = \left(\frac{(S_{in} - S) * Q_{in}}{V} - \sum_{i=1}^n \left[\frac{\mu_{max,i} * S}{K_S + S} - b_{e,i} \right] * \frac{X_i}{Y} \right) \quad \text{SM Eq. 3}$$

$$\frac{dX_i}{dt} = \left(-\frac{Q_{waste}}{V} * X_i \frac{Q_{in} + Q_{recycle}}{Q_{recycle} + Q_{waste}} + \left[\frac{\mu_{max,i} * S}{K_S + S} - b_{e,i} \right] * X_i \right) \quad \text{SM Eq. 4}$$

2099 where Q_{waste} and Q_{recycle} are the waste and recycle flow rates, respectively, and all other parameters are
 2100 defined in the main text. Q_{waste} is determined by the MRT:

$$2101 \quad Q_{\text{waste}} = \frac{V * Q_{\text{recycle}}}{(Q_{\text{in}} + Q_{\text{recycle}}) * \text{MRT} - V} \quad \text{SM Eq. 5}$$

2102 The remaining aspects (e.g., growth parameter combination length, scaling factor, and alpha
 2103 diversity metrics) are handled identically as in the main text SBR model. Using this approach for MRTs of
 2104 1,3,5,7,10, and 15 days, the CSTR model predicted richness values of 22.0, 28.8, 35.5, 41.0, 47.5, and
 2105 54.9 and Shannon diversity indices of 3.08, 3.27, 3.39, 3.45, 3.51, and 3.59, both monotonically increasing
 2106 similar to the SBR model. The full code for the CSTR is include below and can be run by copying into R
 2107 v3.5.1:

```

2108 library(deSolve);packageVersion("deSolve") #At time of run, 1.21
2109 library(matrixStats);packageVersion("matrixStats") #At time of run, 0.54.0
2110 #=====
2111 #Function for Path Length Calculation, taking into consideration ecological limits
2112 #=====
2113 parameter_combination_path_length <- function(MRT, Sstar, Ks, be_constrain, be_eco_min, be_eco_max,
2114 u_constrain, u_eco_max, u_eco_min){
2115   be_max_calc <- c()
2116   be_min_calc <- c()
2117   u_max_calc <- c()
2118   u_min_calc <- c()
2119   for(i in 1:length(MRT)){
2120
2121     if(Sstar[i] < 0){
2122       be_max_calc[i] <- NA
2123       u_max_calc[i] <- NA
2124     } else if( ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] < u_eco_min){
2125       be_max_calc[i] <- NA
2126       u_max_calc[i] <- NA
2127     } else if( ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] > u_eco_max){
2128       be_max_calc[i] <- (Sstar[i]/Ks*(u_eco_max*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))
2129       u_max_calc[i] <- u_eco_max
2130     } else {
2131       be_max_calc[i] <- be_eco_max
2132       u_max_calc[i] <- ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i]
2133     }
2134
2135     if(Sstar[i] < 0){
2136       be_min_calc[i] <- NA
2137       u_min_calc[i] <- NA
2138     } else if( ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] > u_eco_max){
2139       be_min_calc[i] <- NA
2140       u_min_calc[i] <- NA
2141     } else if( ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] < u_eco_min){
2142       be_min_calc[i] <- (Sstar[i]/Ks*(u_eco_min*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))
2143       u_min_calc[i] <- u_eco_min
2144     } else {
2145       be_min_calc[i] <- be_eco_min
2146       u_min_calc[i] <- ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i]
2147     }
2148   }

```

```

2149     }
2150     result <-
2151     list(be_max_calc=be_max_calc,be_min_calc=be_min_calc,u_max_calc=u_max_calc,u_min_calc=u_min_calc)
2152   }
2153
2154   #=====
2155   #Set the ranges of the ecological umax and be values
2156   #=====
2157   u_eco_max <- 9.8
2158   u_eco_min <- 0.2
2159   be_eco_max <- 0.2
2160   be_eco_min <- 0.02
2161
2162   Ks <- 50
2163   #=====
2164   #Set the parameters of the CSTR
2165   #=====
2166   Recycle = 192 # Recycle flow rate
2167   Inflow = 48 # Inflow flow rate
2168   Volume = 12 # Volume of reactor
2169   Slin = 250 # Substrate inflow rate
2170   Ys1S1 = 0.67 # Yield of bacterial growth (fixed) (Henze et al., 1987)
2171   times <- seq(0, 100, by = 1) # Reactor operation time (community equilibrium time)
2172
2173   #=====
2174   #Calculate the waste rate at each MRT based off the parameters and the MRT range
2175   #=====
2176   #Set the MRT
2177   MRT <- c(1,1,3,5,7,10,15)
2178
2179   Waste = Recycle*Volume/(Inflow*MRT+Recycle*MRT-Volume)
2180
2181   #=====
2182   #Set the constraining umax and be values
2183   #=====
2184   u_constrain <- 4.5
2185   be_constrain <- 0.11
2186
2187   Sstar <- Ks*(1+be_constrain*MRT)/((u_constrain-be_constrain)*MRT-1)
2188
2189   #=====
2190   #Calculate the niche path length at each MRT
2191   #=====
2192   path <- parameter_combination_path_length(MRT, Sstar, Ks, be_constrain, be_eco_min, be_eco_max,
2193   u_constrain, u_eco_max, u_eco_min)
2194
2195   # Note: Because the values are normalized above, must back transform into the appropriate actual growth
2196   rate here
2197   Scaling_Factor = 0.25
2198   #=====
2199   # Calculate the normalized distance of the nice path length at each MCRT
2200   #=====
2201   distance <- sqrt( ((path$be_max_calc-
2202   path$be_min_calc)/(path$be_max_calc/2+path$be_min_calc/2)*Scaling_Factor)^2 + ((path$u_max_calc-
2203   path$u_min_calc)/(path$u_max_calc/2+path$u_min_calc/2))^2)
2204
2205   richness <- distance
2206   #=====
2207   # Select MRTs to use in the differential equations, establish certain required variables
2208   #=====
2209   selected_MRT = c(2,3,4,5,6,7)
2210   RW1 <- c()
2211   j <- 1
2212   diversity <- c()
2213   bs <- data.frame()
2214   us <- data.frame()
2215   spike_masses <- c()
2216
2217   #=====
2218   # For each MCRT selected, run the differential equation and determine the Shannon diversity number

```



```

2219 #=====
2220 for (i in selected_MRT){
2221
2222     print(paste("Evaluating MRT # ", i, sep=""))
2223
2224     n = round(distance[i]*50, digits = 0)
2225
2226     print(paste("The number of niches modelled is ", n, sep=""))
2227
2228     urange = seq(path$u_min_calc[i], path$u_max_calc[i], length.out = n)
2229     be = (Sstar[i]/Ks*(urange*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))
2230
2231 #=====
2232 #Establish the differential equations for an CSTR AS system as a string to be parsed
2233 #=====
2234     dX_eq <- paste("dX_", 1:n, " <- (-Qw * X_", 1:n, " * (Qin + Qr)/(Qr + Qw) + u_1_1", 1:n, " *
2235 (So/(Ks + So)) * V * X_", 1:n, " - b_1_1", 1:n, " * V * X_", 1:n, ") / V", sep="")
2236     dmass_eq <- paste("dmass <- ", paste("dX_", 1:n, sep="", collapse=" + "), sep="")
2237     dS_eq <- paste("dSo <- (Qin/V * (SSlin - So) - ", paste("u_1_1", 1:n, " * (So/(Ks + So)) * X_", 1:n, "
2238 / Y_", sep="", collapse=" - "), sep="")
2239     state_list <- paste("list(c(dSo, ", paste("dX_", 1:n, sep="", collapse=" ,"), ", dmass))", sep="",
2240 collapse = "")
2241 #=====
2242 #Parse the differential equations into a function entitles "Lorenz"
2243 #=====
2244     Lorenz <- function(t, state, parameters){
2245         with(as.list(c(state, parameters)),{
2246             eval(parse(text=dX_eq))
2247             eval(parse(text=dmass_eq))
2248             eval(parse(text=dS_eq))
2249             eval(parse(text=state_list))
2250         })
2251     }
2252
2253     parameters500 <- c(b_1_1 = be, Qr=Recycle, Qw=Waste[i], Qin = Inflow, SSlin = Slin, V = Volume,
2254 u_1_1 = urange, Ks = Ks, Y_ = Ys1S1)
2255
2256 #=====
2257 #Set the initial State
2258 #=====
2259     num_spec <- as.numeric(seq(1,1, length.out = n))
2260     state <- c(So = 250, X_ = num_spec, mass = n)
2261
2262 #=====
2263 #Run the differential equation
2264 #=====
2265     RW1 <- ode(y = state, times = times, func = Lorenz, parms = parameters500)
2266
2267 #=====
2268 #Calculate the Shannon diversity metric
2269 #=====
2270     mass_i <- n + 3
2271     final_X <- n + 2
2272     nam <- paste("SRT", i, sep="")
2273     assign(nam, data.frame(be, urange, RW1[99,3:final_X]))
2274     diversity[j] <- sum(-1*RW1[99,3:final_X]/(RW1[99,mass_i]) * log(RW1[99,3:final_X]/(RW1[99,mass_i])))
2275     j = j + 1
2276 #=====
2277 }
2278
2279 MRT[2:7]
2280 richness[2:7]*50
2281 diversity

```

Supplemental Table 1. Sequencing Batch Reactor experimental operation conditions and how the model represents those

Parameter	Experimental Operation	Model Treatment
Innocula	Biomass from the Niederglatt Wastewater Treatment Plant (Niederglatt, Switzerland) aerobic nitrifying reactor	X_0
Substrate	Domestic wastewater	$S_{IN} = 250 \text{ mg/L}$
Reactor Full Volume	12 L	$V_{full} = 12 \text{ L}$
Operation Mode	Sequencing Batch Reactor (SBR)	SBR
Cycle Time	4 hr	4 hr
Hydraulic Residence Time	12 hr	A total of 4 L of reactor volume is removed per cycle
Sequencing Batch Reactor Cycle	Fill: 4 L added over 20 min	$Q_{fill} = 12 \text{ L/hr}$, $t_{fill} = 0.333 \text{ hr}$
	React: 3 hrs of bubbled aeration with mixing	$t_{react} = 3 \text{ hr}$
	Mixed Drain: Drain volume of mixed sludge to achieve the MRT*. This occurs over 20 minutes within the react cycle	$V_{mixed \text{ drain}} = \text{variable}$, $Q_{mixed \text{ drain}} = V_{mixed \text{ drain}}/t_{mixed \text{ drain}}$, $t_{mixed \text{ drain}} = 0.333 \text{ hr}$
	Settle: 30 min of settling without aeration	$t_{settle} = 0.5 \text{ hr}$
	Clarified Drain: The clarified water is drained to achieve a total of 4 L removed	$Q_{clarified \text{ drain}} = (4 - V_{mixed \text{ drain}})/t_{clarified \text{ drain}}$, $t_{clarified \text{ drain}} = 0.333 \text{ hr}$
Microbial Residence Time (MRT*)	1 d : 2 L $V_{mixed \text{ drain}}$ each cycle	$V_{mixed \text{ drain}} = 2 \text{ L}$ each cycle
	3 d : 2 L $V_{mixed \text{ drain}}$ every other cycle	$V_{mixed \text{ drain}} = 2 \text{ L}$ every other cycle
	5 d : 1.2 L $V_{mixed \text{ drain}}$ every third cycle	$V_{mixed \text{ drain}} = 1.2 \text{ L}$ every third cycle
	7 d : 1.7 L $V_{mixed \text{ drain}}$ every sixth cycle	$V_{mixed \text{ drain}} = 1.7 \text{ L}$ every sixth cycle
	10 d : 1.2 L $V_{mixed \text{ drain}}$ every sixth cycle	$V_{mixed \text{ drain}} = 1.2 \text{ L}$ every sixth cycle
	15 d : 0.8 L $V_{mixed \text{ drain}}$ every sixth cycle	$V_{mixed \text{ drain}} = 0.8 \text{ L}$ every sixth cycle
Temperature	Ambient	Not Considered
pH	Experimental Time-point 1: 7.5-8.0 and 2: 7.9-8.3	Not Considered

2282
2283

Supplemental Table 2. 16S rRNA Primer Details

ID	Link Fwd	Primer Sequence Fwd (5'-3')	Link Rev	Primer Sequence Rev (5'-3')	<i>E. coli</i> ID	Size (bp)	Regions
B1	CA	AGAGTTTGATCMTGGCTCAG	TG	ATTACCGCGGCTGCTGG	27f-517r	431-506	V1-V3
B2	AT	GTGCCAGCMGCCGCGGTAA	AC	GGACTACHVGGGTWTCTAAT	515f-806R	252-254	V3-V4

2284

Supplemental Table 3. 16S rRNA B1 Primer Library Results

Library ID	Sample Id	Raw	Merged	Primer	Clean	MeanLength	% Surviving	Assigned to ESVs	% Assigned to ESVs
B1cR1-1_Exp1	TP1 1d cDNA 1	148840	128839	126409	124590	475.933	83.71	109357	87.77
B1cR1-2_Exp1	TP1 1d cDNA 2	141997	123321	121074	119356	475.946	84.06	104922	87.91
B1cR1-3_Exp1	TP1 1d cDNA 3	136694	117406	115087	113457	475.737	83.00	100664	88.72
B1cR2-1_Exp1	TP1 3d cDNA 1	118879	103397	101475	100172	471.485	84.26	89776	89.62
B1cR2-2_Exp1	TP1 3d cDNA 2	167028	145179	142519	140757	471.908	84.27	128911	91.58
B1cR2-3_Exp1	TP1 3d cDNA 3	135679	117235	115034	113588	471.43	83.72	103549	91.16
B1cR3-1_Exp1	TP1 5d cDNA 1	109772	93751	92017	90838	472.117	82.75	80906	89.07
B1cR3-2_Exp1	TP1 5d cDNA 2	170236	144327	141610	139970	472.588	82.22	127562	91.14
B1cR3-3_Exp1	TP1 5d cDNA 3	140345	118632	116268	114810	472.277	81.81	105755	92.11
B1cR4-1_Exp1	TP1 7d cDNA 1	154534	130674	128308	126464	477.025	81.84	114199	90.30
B1cR4-2_Exp1	TP1 7d cDNA 2	147855	124450	122201	120539	477.581	81.53	112035	92.95
B1cR4-3_Exp1	TP1 7d cDNA 3	142932	117563	115226	113731	477.534	79.57	105906	93.12
B1cR5-1_Exp1	TP1 10d cDNA 1	102678	85893	84305	83061	479.619	80.89	77349	93.12
B1cR5-2_Exp1	TP1 10d cDNA 2	183737	152123	149373	147212	479.881	80.12	138011	93.75
B1cR5-3_Exp1	TP1 10d cDNA 3	148051	121778	119459	117684	479.833	79.49	110547	93.94
B1cR6-1_Exp1	TP1 15d cDNA 1	105169	85080	83469	82458	479.767	78.41	77182	93.60
B1cR6-2_Exp1	TP1 15d cDNA 2	168924	141298	137177	136667	479.762	80.90	129372	94.66
B1cR6-3_Exp1	TP1 15d cDNA 3	152302	124288	121892	120191	479.72	78.92	113557	94.48
B1gR1-1_Exp1	TP1 1d gDNA 1	177990	152006	149110	143929	468.946	80.86	138142	95.98
B1gR1-2_Exp1	TP1 1d gDNA 2	173932	150198	147317	142366	468.978	81.85	137153	96.34
B1gR1-3_Exp1	TP1 1d gDNA 3	179171	149928	146977	142904	468.671	79.76	136739	95.69
B1gR2-1_Exp1	TP1 3d gDNA 1	187863	168976	165937	164507	451.081	87.57	161137	97.95
B1gR2-2_Exp1	TP1 3d gDNA 2	170712	153803	150907	149561	451.273	87.61	146489	97.95
B1gR2-3_Exp1	TP1 3d gDNA 3	169384	150136	147267	146150	451.248	86.28	142808	97.71
B1gR3-1_Exp1	TP1 5d gDNA 1	143402	110670	108569	108120	450.413	75.40	103698	95.91
B1gR3-2_Exp1	TP1 5d gDNA 2	190782	165792	162767	161876	450.634	84.85	157856	97.52
B1gR3-3_Exp1	TP1 5d gDNA 3	148960	131653	129278	128519	450.931	86.28	125231	97.44
B1gR4-1_Exp1	TP1 7d gDNA 1	212179	185105	181639	180040	455.078	84.85	175021	97.21
B1gR4-2_Exp1	TP1 7d gDNA 2	164239	144878	142124	140921	455.298	85.80	137339	97.46
B1gR4-3_Exp1	TP1 7d gDNA 3	167027	144444	141691	140429	455.397	84.08	136287	97.05
B1gR5-1_Exp1	TP1 10d gDNA 1	144204	62103	60718	60297	459.615	41.81	50646	83.99
B1gR5-2_Exp1	TP1 10d gDNA 2	129958	107649	105550	104406	459.513	80.34	100654	96.41
B1gR5-3_Exp1	TP1 10d gDNA 3	164147	140554	137992	136383	459.588	83.09	131843	96.67
B1gR6-1_Exp1	TP1 15d gDNA 1	176759	154957	152232	150323	459.807	85.04	145376	96.71
B1gR6-2_Exp1	TP1 15d gDNA 2	133794	116622	114548	113345	460.064	84.72	109667	96.76
B1gR6-3_Exp1	TP1 15d gDNA 3	119478	101326	99473	98440	459.871	82.39	94689	96.19
B1pos-1_Exp1	TP1 Positive	122759	96793	94928	92150	486.235	75.07	88269	95.79
B1pos-2_Exp1	TP1 Positive	59922	50863	49802	46664	486.687	77.87	45404	97.30
B1pos-3_Exp1	TP1 Positive	163606	124395	121882	117927	486.128	72.08	112787	95.64
B1cR1-1_Exp2	TP2 1d cDNA 1	91448	81760	80344	80317	476.023	87.83	66833	83.21
B1cR1-2_Exp2	TP2 1d cDNA 2	73627	65807	64684	64656	476.185	87.82	54271	83.94
B1cR1-3_Exp2	TP2 1d cDNA 3	69696	62343	61305	61287	476.138	87.93	51168	83.49
B1cR2-1_Exp2	TP2 3d cDNA 1	47866	42909	42195	42181	476.6	88.12	35854	85.00
B1cR2-2_Exp2	TP2 3d cDNA 2	51369	45966	45219	45204	476.914	88.00	38786	85.80
B1cR2-3_Exp2	TP2 3d cDNA 3	51594	45861	45086	45068	476.859	87.35	38638	85.73
B1cR3-1_Exp2	TP2 5d cDNA 1	61916	54849	53917	53893	476.216	87.04	46062	85.47
B1cR3-2_Exp2	TP2 5d cDNA 2	75099	66602	65461	65435	476.568	87.13	55908	85.44
B1cR3-3_Exp2	TP2 5d cDNA 3	55735	49162	48296	48285	476.383	86.63	40793	84.48
B1cR4-1_Exp2	TP2 7d cDNA 1	79455	70843	69626	69610	477.414	87.61	60671	87.16
B1cR4-2_Exp2	TP2 7d cDNA 2	68544	61081	60051	60042	477.407	87.60	52418	87.30
B1cR4-3_Exp2	TP2 7d cDNA 3	56380	41693	41008	40999	477.3	72.72	34182	83.37
B1cR5-1_Exp2	TP2 10d cDNA 1	84086	75519	74202	74189	477.927	88.23	63169	85.15
B1cR5-2_Exp2	TP2 10d cDNA 2	77703	69307	68117	68104	477.86	87.65	57918	85.04
B1cR5-3_Exp2	TP2 10d cDNA 3	70196	60512	59457	59442	477.802	84.68	50538	85.02
B1cR6-1_Exp2	TP2 15d cDNA 1	81207	71679	70410	70398	479.604	86.69	60701	86.23
B1cR6-2_Exp2	TP2 15d cDNA 2	68524	60301	59255	59245	479.64	86.46	51536	86.99
B1cR6-3_Exp2	TP2 15d cDNA 3	74221	65275	64159	64146	479.606	86.43	55289	86.19
B1gR1-1_Exp2	TP2 1d gDNA 1	80308	71352	70082	70032	473.979	87.20	57137	81.59
B1gR1-2_Exp2	TP2 1d gDNA 2	83048	73418	72056	72005	474.101	86.70	59123	82.11
B1gR1-3_Exp2	TP2 1d gDNA 3	82328	70791	69556	69510	473.922	84.43	57705	83.02
B1gR2-1_Exp2	TP2 3d gDNA 1	59396	53146	52140	51980	471.11	87.51	45329	87.20
B1gR2-2_Exp2	TP2 3d gDNA 2	60586	53809	52812	52631	471.372	86.87	46197	87.78
B1gR2-3_Exp2	TP2 3d gDNA 3	80581	72097	70767	70613	471.131	87.63	62666	88.75
B1gR3-1_Exp2	TP2 5d gDNA 1	49581	43861	43068	42993	470.77	86.71	39200	91.18
B1gR3-2_Exp2	TP2 5d gDNA 2	68528	60695	59491	59389	470.609	86.66	54881	92.41
B1gR3-3_Exp2	TP2 5d gDNA 3	75799	66785	65578	65464	470.693	86.37	60132	91.86
B1gR4-1_Exp2	TP2 7d gDNA 1	65558	58111	57046	57017	476.276	86.97	51122	89.66
B1gR4-2_Exp2	TP2 7d gDNA 2	25288	9991	9767	9760	475.355	38.60	7152	73.28
B1gR4-3_Exp2	TP2 7d gDNA 3	63554	56358	55325	55299	476.32	87.01	49953	90.33

2285
2286

B1gR5-1_Exp2	TP2 10d gDNA 1	86415	77496	76084	76044	475.843	88.00	69804	91.79
B1gR5-2_Exp2	TP2 10d gDNA 2	72367	61713	60601	60537	475.881	83.65	54937	90.75
B1gR5-3_Exp2	TP2 10d gDNA 3	93512	83575	82082	82014	476.003	87.70	75956	92.61
B1gR6-1_Exp2	TP2 15d gDNA 1	76440	67028	65793	65737	476.039	86.00	59719	90.85
B1gR6-2_Exp2	TP2 15d gDNA 2	65236	57340	56315	56278	475.922	86.27	51164	90.91
B1gR6-3_Exp2	TP2 15d gDNA 3	76133	65492	64354	64301	475.737	84.46	58488	90.96
B1-NEG_Exp2	TP2 Negative	68	25	12	12	444.769	17.65	12	100.00
Total		8,379,312	7,132,637	7,000,872	6,928,919		82.69	6,382,207	92.11

Supplemental Table 4. 16S rRNA B2 Primer Library Results

Library ID	Sample Id	Raw	Merged	Primer	Clean	MeanLength	% Surviving	Assigned to ESVs	% Assigned to ESVs
B2cR1-1_Exp1	TP1 1d cDNA 1	111057	107434	105685	105685	253.0	95.16	104323	98.71
B2cR1-2_Exp1	TP1 1d cDNA 2	94108	91064	89602	89602	253.0	95.21	88466	98.73
B2cR1-3_Exp1	TP1 1d cDNA 3	100513	97253	95625	95624	253.0	95.14	94274	98.59
B2cR2-1_Exp1	TP1 3d cDNA 1	96899	93809	92311	92309	253.0	95.26	91146	98.74
B2cR2-2_Exp1	TP1 3d cDNA 2	147762	143384	140941	140939	253.0	95.38	139098	98.69
B2cR2-3_Exp1	TP1 3d cDNA 3	97671	94850	93243	93240	253.0	95.46	92086	98.76
B2cR3-1_Exp1	TP1 5d cDNA 1	91738	88776	87316	87311	253.0	95.17	86356	98.91
B2cR3-2_Exp1	TP1 5d cDNA 2	145781	140827	138661	138657	253.0	95.11	137297	99.02
B2cR3-3_Exp1	TP1 5d cDNA 3	112221	108883	107133	107130	253.0	95.46	106055	99.00
B2cR4-1_Exp1	TP1 7d cDNA 1	102928	99605	98039	98036	253.0	95.25	97132	99.08
B2cR4-2_Exp1	TP1 7d cDNA 2	127090	122334	120398	120393	253.0	94.73	119325	99.11
B2cR4-3_Exp1	TP1 7d cDNA 3	114762	112162	110393	110390	253.0	96.19	109491	99.19
B2cR5-1_Exp1	TP1 10d cDNA 1	117206	112043	110308	110304	253.0	94.11	109527	99.30
B2cR5-2_Exp1	TP1 10d cDNA 2	138460	133306	131396	131387	253.0	94.89	130558	99.37
B2cR5-3_Exp1	TP1 10d cDNA 3	125783	121311	119362	119357	253.0	94.89	118632	99.39
B2cR6-1_Exp1	TP1 15d cDNA 1	138337	132485	130224	130212	253.0	94.13	129225	99.24
B2cR6-2_Exp1	TP1 15d cDNA 2	175836	168618	166069	166055	253.0	94.44	164836	99.27
B2cR6-3_Exp1	TP1 15d cDNA 3	127849	123203	121245	121235	253.0	94.83	120368	99.28
B2gR1-1_Exp1	TP1 1d gDNA 1	131698	117830	110826	110806	253.1	84.14	110281	99.53
B2gR1-2_Exp1	TP1 1d gDNA 2	99830	89948	84411	84396	253.1	84.54	83962	99.49
B2gR1-3_Exp1	TP1 1d gDNA 3	113347	103711	97964	97953	253.1	86.42	97456	99.49
B2gR2-1_Exp1	TP1 3d gDNA 1	139965	129019	121991	121964	253.0	87.14	121597	99.70
B2gR2-2_Exp1	TP1 3d gDNA 2	117613	106626	99848	99828	253.0	84.88	99468	99.64
B2gR2-3_Exp1	TP1 3d gDNA 3	125997	116938	109707	109697	253.0	87.06	109348	99.68
B2gR3-1_Exp1	TP1 5d gDNA 1	141247	131231	124275	124244	253.0	87.96	123827	99.66
B2gR3-2_Exp1	TP1 5d gDNA 2	117368	109572	103850	103825	253.0	88.46	103428	99.62
B2gR3-3_Exp1	TP1 5d gDNA 3	127142	120328	114432	114409	253.0	89.99	114017	99.66
B2gR4-1_Exp1	TP1 7d gDNA 1	144396	133668	124660	124643	253.1	86.32	124181	99.63
B2gR4-2_Exp1	TP1 7d gDNA 2	109825	103071	96516	96503	253.1	87.87	96100	99.58
B2gR4-3_Exp1	TP1 7d gDNA 3	119476	112044	105121	105101	253.1	87.97	104700	99.62
B2gR5-1_Exp1	TP1 10d gDNA 1	159643	147429	138244	138224	253.1	86.58	137772	99.67
B2gR5-2_Exp1	TP1 10d gDNA 2	138183	126399	117728	117701	253.1	85.18	117272	99.64
B2gR5-3_Exp1	TP1 10d gDNA 3	126324	117713	110205	110193	253.1	87.23	109823	99.66
B2gR6-1_Exp1	TP1 15d gDNA 1	161449	148695	139643	139624	253.1	86.48	139021	99.57
B2gR6-2_Exp1	TP1 15d gDNA 2	121930	112455	104549	104537	253.1	85.74	104090	99.57
B2gR6-3_Exp1	TP1 15d gDNA 3	120103	112458	105560	105542	253.1	87.88	105095	99.58
B2pos-1_Exp1	TP1 Positive	125673	117045	114327	114313	252.7	90.96	113946	99.68
B2pos-2_Exp1	TP1 Positive	174729	162894	159177	159158	252.7	91.09	158605	99.65
B2pos-3_Exp1	TP1 Positive	1681597	1566142	1527964	1527771	252.7	90.85	1524061	99.76
B2cR1-1_Exp2	TP2 1d cDNA 1	103801	101585	99747	99747	253.0	96.09	98377	98.63
B2cR1-2_Exp2	TP2 1d cDNA 2	83785	81924	80507	80506	253.0	96.09	79466	98.71
B2cR1-3_Exp2	TP2 1d cDNA 3	72522	71483	70243	70243	253.0	96.86	69465	98.89
B2cR2-1_Exp2	TP2 3d cDNA 1	86889	85385	83806	83805	253.0	96.45	82804	98.81
B2cR2-2_Exp2	TP2 3d cDNA 2	81494	80036	78542	78542	253.0	96.38	77606	98.81
B2cR2-3_Exp2	TP2 3d cDNA 3	64013	63058	61949	61949	253.0	96.78	61262	98.89
B2cR3-1_Exp2	TP2 5d cDNA 1	119412	117342	115329	115326	253.0	96.58	113962	98.82
B2cR3-2_Exp2	TP2 5d cDNA 2	127474	125088	122854	122853	253.0	96.37	121455	98.86
B2cR3-3_Exp2	TP2 5d cDNA 3	120815	119076	116953	116951	253.0	96.80	115564	98.81
B2cR4-1_Exp2	TP2 7d cDNA 1	119252	117218	115164	115163	253.1	96.57	113990	98.98
B2cR4-2_Exp2	TP2 7d cDNA 2	107022	104924	103074	103073	253.1	96.31	101990	98.95
B2cR4-3_Exp2	TP2 7d cDNA 3	83460	82194	80798	80796	253.1	96.81	80054	99.08
B2cR5-1_Exp2	TP2 10d cDNA 1	110965	109021	107033	107033	253.0	96.46	105832	98.88
B2cR5-2_Exp2	TP2 10d cDNA 2	94767	93159	91497	91497	253.1	96.55	90525	98.94
B2cR5-3_Exp2	TP2 10d cDNA 3	107829	106427	104446	104445	253.0	96.86	103393	98.99
B2cR6-1_Exp2	TP2 15d cDNA 1	86686	85152	83692	83692	253.1	96.55	82904	99.06
B2cR6-2_Exp2	TP2 15d cDNA 2	191897	188630	185333	185331	253.1	96.58	183463	98.99
B2cR6-3_Exp2	TP2 15d cDNA 3	106495	104983	103090	103090	253.1	96.80	102136	99.07
B2gR1-1_Exp2	TP2 1d gDNA 1	72516	70523	68169	68169	253.0	94.01	67510	99.03
B2gR1-2_Exp2	TP2 1d gDNA 2	62366	60587	58369	58369	253.0	93.59	57867	99.14
B2gR1-3_Exp2	TP2 1d gDNA 3	75284	72867	70302	70302	253.0	93.38	69594	98.99
B2gR2-1_Exp2	TP2 3d gDNA 1	73891	71416	68528	68528	253.0	92.74	68113	99.39
B2gR2-2_Exp2	TP2 3d gDNA 2	59311	57211	54864	54864	253.0	92.50	54554	99.43
B2gR2-3_Exp2	TP2 3d gDNA 3	72605	69802	66834	66834	253.0	92.05	66429	99.39
B2gR3-1_Exp2	TP2 5d gDNA 1	103791	100694	97264	97263	253.0	93.71	96743	99.47
B2gR3-2_Exp2	TP2 5d gDNA 2	83375	80910	77972	77972	253.0	93.52	77601	99.52
B2gR3-3_Exp2	TP2 5d gDNA 3	109645	106096	102160	102158	253.0	93.17	101566	99.42
B2gR4-1_Exp2	TP2 7d gDNA 1	111611	107887	103392	103383	253.0	92.63	102819	99.45
B2gR4-2_Exp2	TP2 7d gDNA 2	117661	113613	108643	108633	253.0	92.33	108143	99.55
B2gR4-3_Exp2	TP2 7d gDNA 3	127352	122909	117844	117838	253.0	92.53	117186	99.45
B2gR5-1_Exp2	TP2 10d gDNA 1	125615	120989	115704	115703	253.1	92.11	115040	99.43
B2gR5-2_Exp2	TP2 10d gDNA 2	109372	105588	101001	101001	253.1	92.35	100505	99.51
B2gR5-3_Exp2	TP2 10d gDNA 3	107330	103461	99039	99038	253.1	92.27	98450	99.41
B2gR6-1_Exp2	TP2 15d gDNA 1	71655	68957	65926	65925	253.0	92.00	65484	99.33
B2gR6-2_Exp2	TP2 15d gDNA 2	118048	113599	108541	108540	253.0	91.95	107904	99.41
B2gR6-3_Exp2	TP2 15d gDNA 3	131629	126300	120627	120627	253.0	91.64	119804	99.32
B2-NEG_Exp2	TP2 Negative	174	91	17	14	236.1	8.05	12	85.71

cNegA_Exp2	TP2 Negative	839	619	585	582	252.6	69.37	571	98.11
cNegB_Exp2	TP2 Negative	114	28	17	16	238.2	14.04	16	100.00
gM1_Exp2	TP1 Positive	67197	65911	64138	64130	253.0	95.44	63995	99.79
gM2_Exp2	TP1 Positive	48324	47469	46306	46301	253.0	95.81	46184	99.75
gNegA_Exp2	TP2 Negative	177	68	43	37	246.5	20.90	31	83.78
gNegB_Exp2	TP2 Negative	109	18	7	4	202.4	3.67	4	100.00
Total		10,182,105	9,700,861	9,389,298	9,388,571		92.21	9326618	99.34

2288

Supplemental Table 5. The intermediate processing results of the mRNA annotation pipeline

Sample	Input Reads	Quality Filtered Reads	Pass Quality Reads	% Pass Quality	rRNA Filtered	Non- Filtered (Surviving)	% Surviving Overall	EC Annotated Reads	% Surviving Reads EC Annotated	% Surviving Reads Uniprot- TrEMBL Annotated	Normalization Factor
TP1 1d	51659728	857629	50802099	98.3	7040898	43761201	84.7	9829704	22.5	48.6	2127249485
TP1 3d	41274968	855370	40419598	97.9	7537452	32882146	79.7	5602624	17	38.1	1252836068
TP1 5d	43478158	952651	42525507	97.8	6561606	35963901	82.7	5104415	14.2	31.5	1133057087
TP1 7d	43956036	789388	43166648	98.2	5201824	37964824	86.4	6811803	17.9	38.2	1449679211
TP1 10d	43862862	877647	42985215	98	4458080	38527135	87.8	6253848	16.2	33.8	1302949179
TP1 15d	54367240	1128268	53238972	97.9	5545211	47693761	87.7	7341422	15.4	31.7	1512703018
TP2 1d	49296213	786470	48509743	98.4	11095152	37414591	75.9	7186540	19.2	40.7	1521307202
TP2 3d	51239482	896696	50342786	98.2	13212030	37130756	72.5	6027312	16.2	36.2	1343769486
TP2 5d	41869186	738365	41130821	98.2	6685061	34445760	82.3	5725054	16.6	38.8	1337384189
TP2 7d	51174390	998022	50176368	98	9662159	40514209	79.2	6677636	16.5	36	1457150247
TP2 10d	47099688	978073	46121615	97.9	4918694	41202921	87.5	6990419	17	35.4	1457520368
TP2 15d	47708399	919600	46788799	98.1	8545328	38243471	80.2	6114888	16	33	1260275343

2289

Supplemental Table 6. Results of the Sequencing Batch Reactor multi-species Monod model

MRT	Maximum μ_{\max} (day ⁻¹)	Minimum μ_{\max} (day ⁻¹)	Richness	Number of Niches	Shannon Diversity	TSS (mg/L)	Substrate Uptake Rate (mg-s/[mg-x day])	k_{theo} (day ⁻¹)
1	5.37	4.63	0.26	13	2.56	141	15.37	5.00
3	5.95	4.05	0.32	16	2.76	397	15.59	5.09
5	6.40	3.60	0.38	19	2.90	561	16.04	5.24
7	6.71	3.29	0.44	22	3.02	668	16.04	5.38
10	7.09	2.91	0.50	25	3.11	795	17.06	5.59
15	7.51	2.49	0.58	29	3.20	902	17.83	5.84

2290